The USENIX Association Magazine

;login:

Windows NT Special Issue

NOVEMBER 1997

Special Issue on Windows NT

USENIX The Advanced Computing Systems Association

upcoming events

USENIX Symposium on Internet Technologies and Systems

WHEN	WHERE	WHO *program chair*
December 8-11/97	Monterey, CA	**Carl Staelin** Hewlett-Packard Laboratories

7th USENIX Security Symposium

Sponsored by USENIX in cooperation with the CERT Coordination Center

WHEN	WHERE	WHO
January 26-29/98	San Antonio, TX	**Avi Rubin**, Program Chair AT&T Labs - Research **Greg Rose**, Invited Talks Coord Qualcomm Australia

DEADLINES

Final Papers
December 9/97

New Network Technologies Symposium

WHEN	WHERE
March 2-3/98	Salt Lake City, UT

NOTE: Cancelled due to lack of submissions.

4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)

WHEN	WHERE	WHO *program chair*
April 27-30/98	Santa Fe, NM	**Joe Sventek** Hewlett-Packard

DEADLINES

Tutorial Submissions	Paper Submissions	Notification to Authors	Final Papers
December 2/97	December 2/97	January 7/98	March 17/98

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 15-19/98	New Orleans, LA	**Fred Douglis**, Program Chair AT&T Labs Invited Talks Coordinators: **Clem Cole** Digital Equipment Corporation **Barry Kercheval** Xerox PARC

DEADLINES

Notification to Authors	Full Papers	Final Papers
January 26/98	March 30/98	April 27/98

2nd USENIX Windows NT Symposium

WHEN	WHERE	WHO *program chair*
August 3-5/98	Seattle, WA	**Thorsten von Eicken** Cornell University **Susan Owicki** Intertrust, Inc.

DEADLINES

Paper Submissions	Break-Out Proposals	Notifications of Acceptance	Final Papers
March 3/98	March 20/98	April 3/98	June 18/98

Large Installation Systems Administration of Windows NT Conference

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO *program co-chairs*
August 5-7/98	Seattle, WA	**Remy Evard** Argonne National Laboratory **Ian Reddy** Simon Fraser University

DEADLINES

Paper Submissions	All Other Submissions	Notification to Authors	Final Papers
March 3/98	March 10/98	March 31/98	June 18/98

3rd USENIX Workshop on Electronic Commerce

WHEN	WHERE	WHO *program chair*
August 31-Sept. 3/98	Boston, MA	**Bennet Yee** University of California, San Diego

DEADLINES

Extended Abstracts	Notifications to Authors	Final Papers
March 6/98	April 17/98	July 21/98

6th Annual Tcl/Tk Conference

WHEN	WHERE	WHO *program co-chairs*
September 14-18/98	San Diego, CA	**Don Libes** NIST **Michael J. McLennan** Bell Labs

DEADLINES

Extended Abtracts	Notification to Authors	Final Papers
March 20/98	April 17/98	July 21/98

12th Systems Administration Conference (LISA '98)

Co-sponsored by USENIX and SAGE, the System Administration Guild

WHEN	WHERE	WHO *program co-chairs*
December 6-11/98	Boston, MA	**Xev Gittler** Lehman Brothers **Rob Kolstad** BSDI, Inc.

contents

;login:

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710

$30 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Editorial Staff

Special Issue Editor:
Rik Farrow <rik@spirit.com>

;login: Editor:
Rob Kolstad <kolstad@usenix.org>

Managing Editor:
Eileen Cohen <cohen@usenix.org>

Copy Editor:
Sylvia Stein Wright

Proofreader:
Kay Keppler

Designer:
Vinje Design

Typesetter:
Festina Lente

Advertising

Zanna Knight <zanna@usenix.org>

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
WWW: <http://www.usenix.org>

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:,
the Association's magazine, published six to eight times a year, featuring technical articles, system administration tips and techniques, practical columns on Perl, Java, and C++, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to papers
from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web <http://www.usenix.org>.

Discounts on registration fees
for all USENIX Conferences, as many as eight every year.

Discounts
on the purchase of proceedings and CD-ROMS from USENIX conferences

PGP Key Signing service
available at conferences.

Discounts
on BSDI, Inc. products.
BSDI information: 800 800 4BSD.

Discounts
on all publications and software from Prime Time Freeware, including Prime Time Freeware for UNIX, Prime Time Freeware for AI, Prime Time TeXcetera, and Tools & Toys for UnixWare.

Discounts
on all publications from The Open Group.

Savings
20% savings on all titles from Prentice Hall PTR <http://www.bookpool.com> and from O'Reilly & Associates (800 998 9938).

Savings (10-20%) on selected titles from McGraw-Hill (212 512 2000), The MIT Press (800 356 0343), Prentice Hall (201 850 6789), Morgan Kaufmann (800 745 7323), and Sage Science Press (805 499 9774)

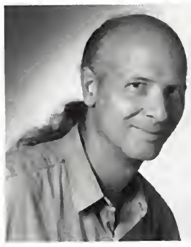
Special subscription rate
to *The Linux Journal* (206 527 3385)

The right to vote
on matters affecting the Association, its bylaws, election of its directors and officers

Optional membership
in SAGE, the System Administrators Guild

For information regarding membership or benefits, please contact
<office@usenix.org>
Phone: 510 528 8649

in this issue . . .



by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*

<rik@spirit.com>

This special edition of *;login:* focuses on Windows NT and the USENIX conferences that took place in Seattle in August. For some of you, the mere mention of NT may be reason enough to shelve or immediately place this magazine in the circular file. But I hope that what we have chronicled here will provide a better understanding of both Microsoft Corporation and its offspring, Windows NT.

The conferences were both close to overbooked – over 300 attendees at each. The focus on a single vendor's product, and the degree of commercialism in some presentations, were very unusual for a USENIX conference. Part of the problem is simply that NT is still the new kid on the block, even though NT, in some form, has been around since 1992.

I got what I wanted: more information about NT, particularly its security. I also learned more about what people are doing to integrate NT into their existing networks, the problems they are having, and how to solve those problems. These solutions appear in the summaries and online versions of the papers <http://www.usenix.org/publications/library/proceedings/usenix-nt97/> and <.../nt-sysadmin97>.

Some of those solutions appear in more detail in this issue. Samba is one of the best UNIX/NT integration solutions, and there are two feature articles about Samba. Dave Korn's U/WIN and Softway's OpenNT provide programming interfaces to NT that will be familiar to any UNIX programmer. In addition, these feature articles helped me understand the different environment faced by programmers moving to NT from UNIX.

The conference also marked the first time I had come face to face with Microsoft, in the form of product managers, engineers, evangelists, and summer interns. I could tell that these people had pride in what they were doing – that they were quite similar to the rest of the software development community. But there was also an edge that was unfamiliar and puzzling to me.

Microsoft dominates the desktop software market and is striving for the server market as well. One Microsoft employee made no bones about this during the conferences, saying, "We want 100% of the market." Not only is this attitude disturbing; it also was proven to be dangerous for the computer giants of the seventies, IBM and DEC.

But there was another aspect of the Microsoft edge that intrigued me. There was a degree of defensiveness that bordered on fear. Why would the most successful software company in the word be afraid of anything? I found a reasonable answer in an unlikely place: *American Heritage Magazine of Inventions and Technology* (Fall 1997), which included an interview with Edward Tenner, a scientific historian and author of *Why Things Bite Back: Technology and the Revenge of Unintended Consequences* (ISBN 0-679-42563-2). Tenner said:

> The difference between Intel and Microsoft and the older giants [AT&T and IBM] is that these organizations have much shorter product cycles, and they have done far

less basic research. They are more sensitive to tremors in the marketplace, more vulnerable potentially to competitors.

Although many people were amazed at the swift change from a position that ignored the Internet to one that embraced it, Microsoft's quick turnaround strongly affirms Tenner's statement. AT&T could spend years designing a telephone that would then have a product life a generation long. And both AT&T and IBM leased their products. They were not relying on a steady stream of sales, but on income from leases.

We learned a lot about NT during the conferences, and Microsoft had a chance to learn from us. The need for command-line interfaces for NT rang particularly loud and clear, to the obvious surprise of the product managers. To me, the goal is a two-way flow of information – to learn about features of NT that are unique and useful and for Microsoft to learn from the many years of experience of programmers and system administrators attending the conferences.

NT is an undocumented operating system, according not only to Microsoft employees, but to source licensees as well. This is because it changes too quickly for documentation to keep up with it. In reality, this means that the interfaces to the operating system, its networking, and authentication mechanisms are secrets. I do not find this comforting, but I am cheered by the efforts of programmers, like the Samba team, which unveil the hidden aspects of NT and help to integrate it with the rest of the computing world.

The conferences revealed more than anything that there is a dearth of detailed knowledge about a major operating system, NT. Microsoft-certified engineers understand GUIs and procedures – they do not have the depth to solve the really difficult problems. Those solutions will come from the members of communities such as USENIX.

Copyright by Peter Wagner. Reprinted with permission of the artist.

implementing an infrastructure framework

by John Holmwood

John Holmwood is an infrastructure architect at NOVA Gas Transmission Ltd. He has been a member of the Calgary UNIX User's Group since 1991 serving on its board of directors from 1992 through 1994. This is his first year as a USENIX member.

<john.holmwood@pipe.nova.ca>

From 1995 until 1997, NOVA Gas Transmission Ltd. (NGTL), a natural gas transmission utility, attempted to build a service-based corporate-distributed computing infrastructure based on Open System Standards. By 1997, they admitted that the attempt had failed. NGTL's information technology group is divided into multiple departments, each responsible for a specific aspect of the computing infrastructure. Each department selected the best products within the restrictions of Open System Standards that met its specific business needs. Unfortunately, the products could not be combined to provide an integrated infrastructure that supported the business.

NGTL Environments

NGTL is the natural gas transmission provider within the province of Alberta, Canada. Physically, its computing infrastructure is made up of a head office in Calgary, Alberta, five regional centers connected to Calgary by a frame relay wide area network, and 11 field offices connected to the regional centers by 56KB wide area networks. The company has 3,000 employees, 2,000 at the Calgary site and the rest spread around the province at the field offices and regional centers. Each employee has a workstation that will be running Microsoft NT by the end of 1997. The Calgary site has a data center where the majority of the company's servers are located. It contains a mainframe and approximately 120 UNIX servers from a variety of vendors. Each regional center and field office will contain Windows NT servers to support the workstations.

NGTL's computing infrastructure is made up of two environments: a *desktop environment*, and a *distributed computing environment.* Figure 1 on the next page shows this infrastructure with the support services.

What's Wrong with Standards

Unfortunately, specifying the standards to which the support services must conform does not ensure that products can be combined to form an integrated framework. This is because standards don't normally define everything about a product. Each company producing a product believes that it must be able to distinguish its product from its competitors.

Consider the PC hardware market. Because the hardware is essentially defined by standards, formal or de facto, the manufacturers must compete with price and quality as the distinguishing characteristics between products. Vendors in other areas of the computing industry do not want to be in this position. Therefore, although the standards define how the various components should interact, each product is slightly different, and products from different vendors will not interface seamlessly. The purchaser is left with the task of making the products work together to provide a consistent infrastructure.

One solution to this problem is to minimize the number of vendors from whom you purchase products. In the ideal world, you could get all of the products you need from

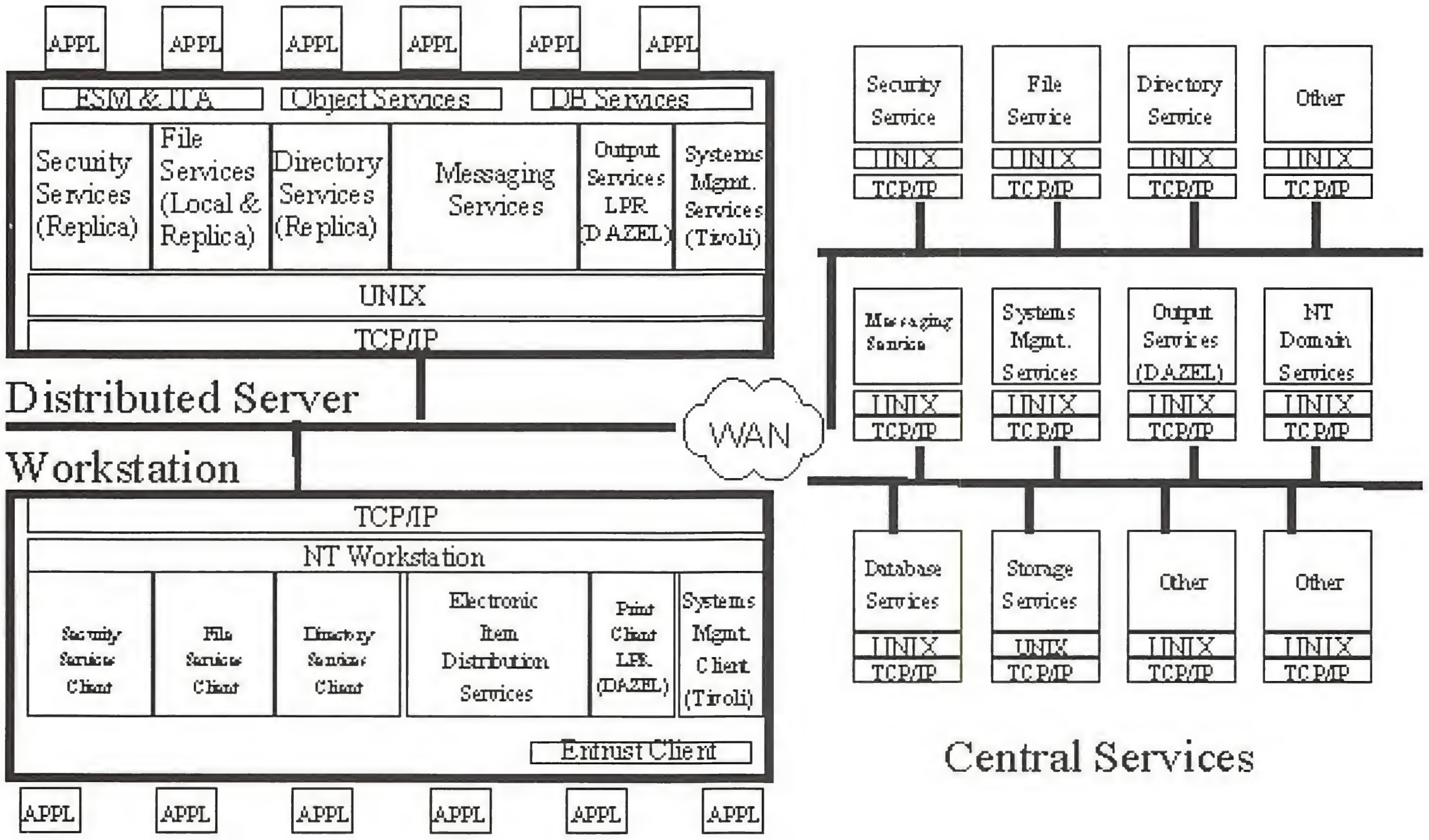


Figure 1: NGTL Infrastructure

one vendor. In fact, this is exactly what each vendor promises. NGTL approached the two vendors with whom it does the most business and asked them to describe their support services framework and how they could provide the services NGTL needed.

Microsoft's Framework

Microsoft holds the dominant position in desktop computing. Its roots are in the personal computing world. In recent years, it has been expanding into enterprise computing; however, its background has resulted in its approaching the enterprise computing space from the point of view of someone using a desktop computer and asking what services are required to support this desktop.

Microsoft's marketing strategy is reminiscent of IBM's strategy of the early 1980s. It produces an integrated line of services that can support small- to medium-sized enterprises. (See <http://www.microsoft.com/vbasic/docs/buscomp.htm>.) The services work very well in a Microsoft-only environment. Part of its strategy appears to be to provide functionality only by combining multiple Microsoft services. Examples of this include the need to deploy SQL Server in order to use SMS or the need for Exchange in order to utilize the workgroup features of Outlook. One result of this portion of Microsoft's strategy is that once Microsoft products get a foothold in your organization, the need for additional Microsoft products seems to proliferate.

Microsoft's strategy results in either duplicate support services, one set for Microsoft products and a second set for everything else, or a custom interface between the Microsoft products and everything else.

For companies with a heterogeneous computing infrastructure, Microsoft's strategy results in either duplicate support services, one set for Microsoft products and a second set for everything else, or a custom interface between the Microsoft products and everything else.

IBM's Framework

IBM is a major player in the enterprise computing environment, particularly with large organizations (those with 1,000 or more users). It approaches the enterprise computing space from the viewpoint of servers and the services they provide. With the Open Blueprint architecture (See <http://www.software.ibm.com/openblue>), IBM has embraced the Open System Standards model for enterprise computing. As could be expected, IBM provides products for almost all of the components in its Open Blueprint. Where IBM doesn't have a product, it specifies third-party products that work seamlessly within the environment. Furthermore, the IBM products work across a range of operating systems, including not only all of IBM's, but also Microsoft's Windows NT and most of the popular versions of UNIX.

The Open Blueprint architecture provides its seamless integration by layering services on top of the operating systems. For example, it uses The Open Group's DCE as the basis of its distributed computing services. This adds significant cost to the infrastructure. Additionally, it doesn't do a good job of integrating the enterprise computing environment with the desktop computing environment dominated by Microsoft. The Open Blueprint's solution is to use World Wide Web-based technologies: HTML and Java.

What NGTL Is Doing

After reviewing the responses from these vendors, NGTL determined that neither vendor could supply all of the support services required. Figure 2 on the next page shows a graphical depiction of NGTL's analysis of the vendors' responses.

An application running on a single computer (whether desktop or mainframe), has more of its support services provided by the operating system than does a distributed application. The result is an ellipse of required support services. As the figure shows, Microsoft provides solutions at the desktop end of the ellipse, and IBM provides solutions at the mainframe end. The section in the middle, where the two frameworks overlap, is where the customer must make choices and develop custom interfaces.

NGTL created an Infrastructure Framework by blending the two vendor frameworks with products already selected for its support services (e.g., Oracle, Entrust, Dazel, etc.). Each vendor's framework addresses one of NGTL's computing environments. The Microsoft framework addresses the desktop environment. The IBM Open Blueprint addresses the distributed computing environment.

Real-World Example

Theory is nice, but to understand how it is being applied, let's look at a real example. NGTL is replacing its mainframe-based groupware system. The system includes email, group scheduling, public folders, and a rudimentary task list. The NGTL framework indicates that the desktop portion of the new system must fit into the Microsoft framework. NGTL has chosen Microsoft Outlook for this component. It integrates smoothly with the productivity applications in the desktop environment and uses the Microsoft protocols for interprocess communications to interact with other applications on the desktop computer. The framework indicates that the messaging backbone that Outlook attaches to should fit into the IBM Open Blueprint framework. This is where problems

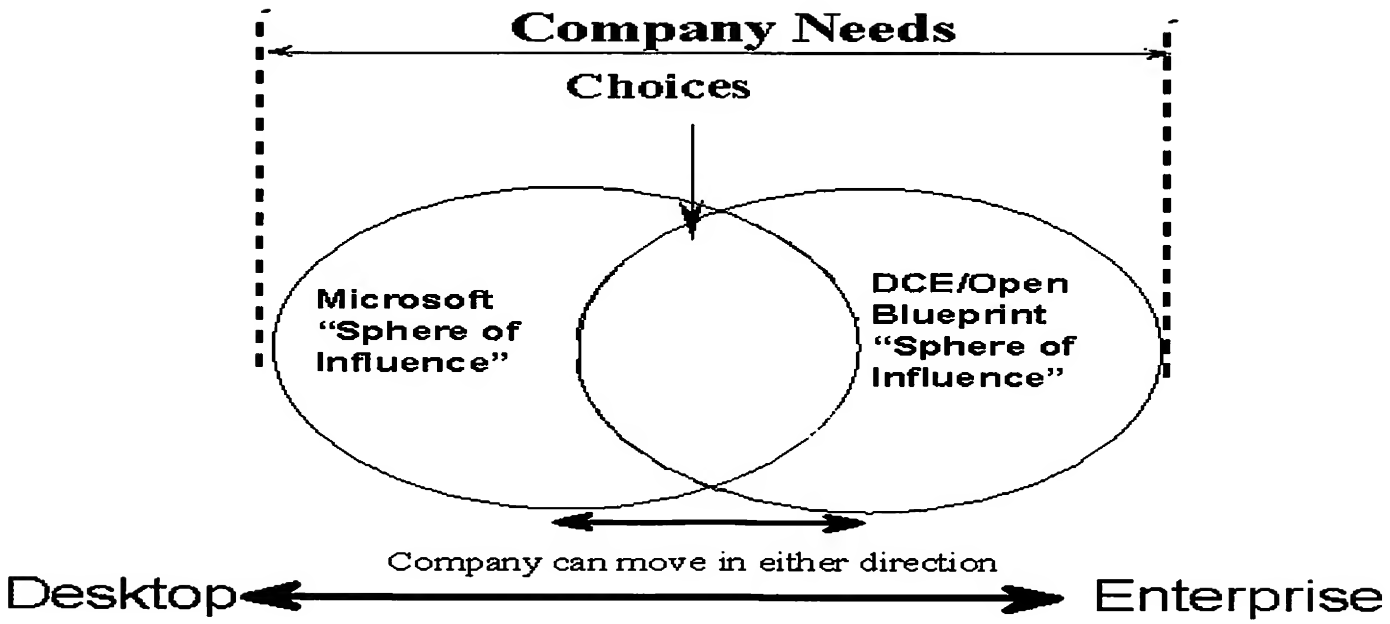

Company Needs
Choices
Microsoft "Sphere of Influence"
DCE/Open Blueprint "Sphere of Influence"
Company can move in either direction
Desktop
Enterprise

Figure 2: NGTL Framework

arise. In order to provide the functionality of the system being replaced, Outlook requires Microsoft Exchange as its messaging backbone. With any other messaging backbone product, Outlook can act only as a personal rather than group scheduler and cannot provide the public folder functions.

Bowing to the inevitable, NGTL is deploying Exchange as its messaging backbone. However, it is being deployed in a manner consistent with the IBM Open Blueprint environment. Each Exchange server is being equipped with the Exchange Internet Gateway product. This allows all other applications and services to access the backbone using the IETF SMTP protocol specified by the IBM framework. Each employee's mail address is specified using the SMTP rather than the X.400 protocol. Internally, Exchange uses the X.400 protocol, but it is hidden away from both the users and the other applications and services.

Messaging is the first and probably simplest integration effort that NGTL has undertaken. Integrating Windows NT into a distributed environment based upon the Open Group's DCE and DFS will be a larger challenge.

Microsoft "embraces and extends" Kerberos V5

by Theodore Ts'o

Theodore Ts'o leads the Kerberos V5 development team at MIT. He is a member of the Internet Engineering Task Force, where he currently serves on the Security Area Directorate and as co-chair of the IP Security Working Group.

<tytso@MIT.EDU>

A lot of excitement was generated by Microsoft's announcement that NT 5.0 would use Kerberos, followed by a lot of controversy when Microsoft announced that it would be adding proprietary extensions to the Kerberos V5 protocol. Exactly what and how Microsoft did and tried to do has been a subject of some confusion. Here's the scoop about what really happened.

NT 5.0 will indeed use Kerberos. However, Microsoft has "embraced and extended" this protocol by adding a digitally signed Privilege Attribute Certificate (PAC) to the Kerberos ticket. The PAC will contain information about the user's 128-bit NT unique id, as well as a list of groups to which the user belongs.

The NT PAC unfortunately is not compatible with the PACs used by the Open Software Foundation's Distributed Computing Environment (DCE). It is also somewhat debatable whether the NT PAC is legal with respect to RFC-1510, the IETF Kerberos V5 protocol specification. The original intent of RFC-1510 prohibited what Microsoft was trying to do, but Microsoft found what it claimed to be a loophole in RFC-1510 specification.

Many folks, including Paul Hill and myself at MIT, as well as Cliff Neumann at ISI, have tried to work with Microsoft to find a more compatible way of doing what they wanted to do. To that end, we made changes in the upcoming revision of RFC-1510 to add a clean and compatible way of adding extensions such as Microsoft's PAC to the Kerberos ticket.

To Microsoft's credit, it agreed to change NT 5.0 to use a cleaner and more compatible way of adding extensions to the Kerberos V5 ticket. It also pledged that it would make available to us detailed technical information about the NT PAC after the beta release of NT 5.0. This pledge was very important to MIT and other commercial, educational, and government sites which have an extensive deployed base of Kerberos V4 applications (for example Transarc's AFS). At MIT, for example, we had planned to add the ability to generate an NT PAC to the MIT Kerberos V5 implementation, which has backwards compatibility for Kerberos V4 applications.

Unfortunately, at the Microsoft Professional Developers Conference (PDC) in September, Microsoft appeared to be backing away from this commitment. For the first time, Microsoft revealed that it had chosen to implement the NT Domain Controller in such a way that the Active Directory Server and the Microsoft KDC run in the same process space, and that NT clients cannot be configured to split a Domain Controller across two machines. Thus, it would not be useful for Microsoft to reveal its proprietary extensions to the Kerberos protocol.

However, at the PDC, Microsoft did indicate that it had licensed its Domain Controller to a few UNIX vendors. So it may eventually be possible to run a Domain Controller on a non-NT machine, but there is no indication what the license may cost each site. It is doubtful, however, whether Kerberos V4 support will be included in those products.

Microsoft should be commended for using a mature industry standard such as Kerberos for its authentication protocol. Kerberos has had a long review period, and its use has been proven in many operational environments. It seems ironic, however, that Microsoft would choose to design and deploy its implementation with features that are guaranteed to alienate the early adopters of Kerberos, the very people that have helped to create and improve the technology that Microsoft has chosen to "embrace and extend."

NT to the max...(NoT)

by Neil Gunther

Neil Gunther is founder and principal consultant for Performance Dynamics Company in Mountain View, CA. Dr. Gunther has worked in the Silicon Valley for 18 years. He is a member of IEEE, ACM, and CMG.

<ngunther@ricochet.net>

To NT or NoT to NT? As someone who analyzes the performance of large-scale UNIX servers, I finally decided to face this nagging question by attending the recent USENIX Windows NT Workshop in Seattle. Seattle's weather being in the 90s all week was as much of a surprise as having the trade names USENIX and NT in the same conference title! Surprises, however, turned out to be a theme of the conference.

Overall, I came away being much more impressed with NT than I had expected, and I'm very glad I attended. But there were some unexpected low points too, and it's one of those I would like to bring to your attention. It has do with a topic near and dear to the hearts of many UNIX sysadmins and others involved in managing large UNIX systems – server scalability [1]. It is also the next frontier for Microsoft.

Naturally, one presentation I was looking forward to was the keynote address by Jim Gray, "Windows NT to the Max – Just How Far Can It Scale Up?" Gray is a respected figure in the database community whose career spans companies like Tandem, DEC, and now Microsoft. Moreover, he's possibly best known as one of the leading evangelists [2] for standardizing database benchmarks that ultimately led to the formation of the Transaction Processing Performance Council (a.k.a. TPC).

Imagine my surprise to hear from Gray that NT could outperform UNIX by scaling to billions of transactions at prices much lower than UNIX platforms. Imagine my dismay at some of the subtle misinformation invoked to make a point! OK. You can't imagine my dismay: either you weren't there or the details were unfamiliar to you. Either way, I would like to examine some points in Gray's presentation with a view to separating the technical "wheat" from the marketing "chaff." Since this will require excursions off the beaten UNIX path, more details will appear in forthcoming *;login:* articles. In this opening salvo, there is only space to highlight the issues I plan to revisit. Eventually, I hope this discussion will better help you understand the scalability of both NT and UNIX.

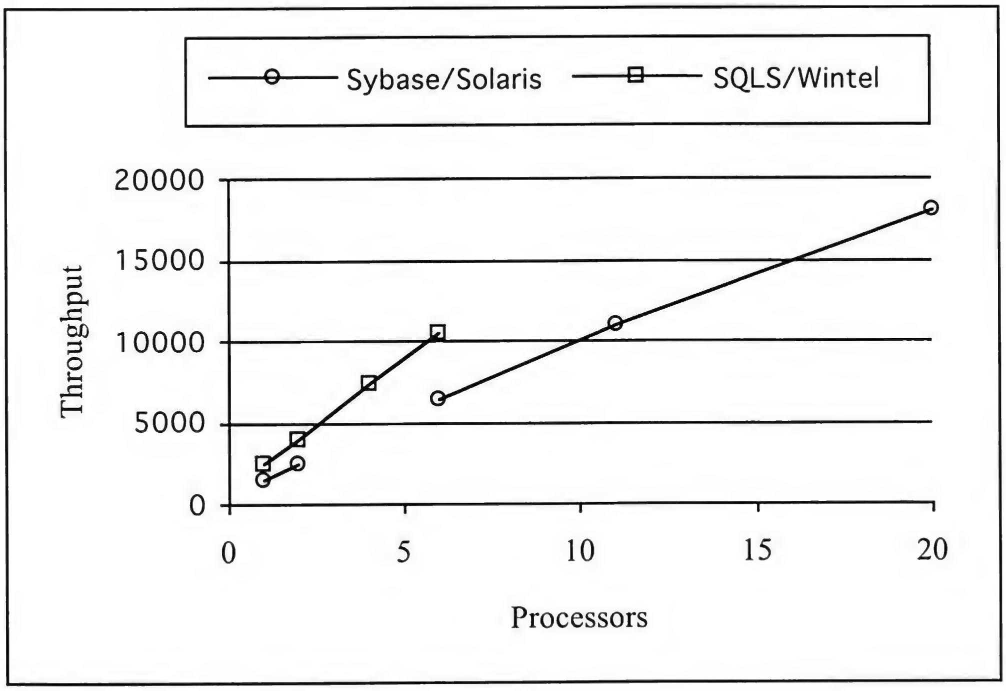
What's Wrong With This Picture?

So, what did the man say? Figure 1 represents OLTP (On-Line Transaction Processing) throughput using the TPC Benchmark-C. The TPC-C benchmark workload models inventory control in a distributed warehouse, and performance is measured in transactions per minute (or tpmC) [3]. The benchmark must be audited by the TPC and documented before being announced publicly; otherwise it's a technical violation of TPC rules.

Figure 1 appeared in Gray's presentation with the title: "NT Scales Better Than Solaris." Although not a usage violation of TPC data, the reader needs to exercise extreme caution when reading comparative charts of this type. The major variables in any TPC benchmark are:

- Platform (e.g., Intel or Sun processors or disks)
- Operating system (e.g., NT or Solaris)
- RDBMS (e.g., database management software; SQL Server or Sybase)
- Application (e.g., the TPC-C benchmarking workload)

The performance analyst's golden rule is: Only change one thing at a time [1]. In Figure 1 there are many things that are different.



Here are some changed variables to watch out for:

- Are all SQLServer data points measured on the same Wintel architecture?
- SQLServer and Sybase are not the same RDBMS.
- Are there any Sybase results on Wintel platforms, and how do they compare?
- Microsoft has full control over SQLServer code and its performance.
- Why are there no TPC-C data points above a 6-way Wintel server?
- Do other RDBMSs scale better than SQLServer on Wintel platforms?
- How contemporaneous were these measurements?

Incidentally, Gray publicly blamed Intel-based hardware for limiting scalability performance above 6-way servers. We'll return to these points in a subsequent article.

Billions and Billions!

To give some idea of the future potential of large-scale NT servers, Gray reported on several scalability projects:

- Online Atlas (1.1 million US place names with SPIN-3 images)
- Tandem's "Two-Ton" (DSS workload on 64 CPUs with 2 TBytes on 480 disks)
- Compaq "Debit-Credit" (OLTP workload on 140 CPUs and 900 disks)

The last of these projects apparently supports over 1 billion database transactions per day. These large-scale prototype systems are nontrivial to construct and the results in themselves are very impressive. Note that 1 billion transactions per day is about 700,000 transactions per minute. But Microsoft has apparently entered the Carl Sagan zone because these are not TPC-C transactions or any other TPC benchmark transactions. Why not? Since Gray did not explain this point to the audience, I asked him about it afterward. He told me they couldn't publish a TPC-C benchmark because SQLServer failed to handle "transparency" (a TPC benchmarking technical requirement).

Clearly, the intended message for the audience was that SQLServer is more than capable of exhibiting superior TPC-like benchmark performance; they just couldn't hack an official TPC number because of an annoying technicality. This is precisely the kind of self-promotional clandestine benchmarking that TPC was formed to discourage.

There is another problem in using a debit-credit workload. As the name indicates, debit-credit applies a simple ATM banking transaction like the TPC-A benchmark (now defunct), not the more complex transaction of TPC-C. A rule of thumb states that TPC-A throughput is about five times greater than TPC-C throughput [1]. One might guess that the Compaq/OLTP throughput would be more like 100,000 TPC-C equivalent transactions per minute. For historical reference, Tandem reported 20,000 tpmC over three years ago on a 120-node MIPS-R4000 Himalaya server.

Cluster (un)Availability

Attracted by the desire to support billions of desktop clients, Gray emphasized Microsoft's focus on cluster technologies. The essential idea is to strap multiple servers together using a high-performance interconnect network to enable both scalable performance and reliability (i.e., no single point of failure – see [1]). But the commercial cluster concept is not new.

There are several historical precedents for scalable clusters that support commercial database workloads: in particular, Tandem for OLTP, Teradata for DSS (Decision Support Systems), and the IBM Parallel Sysplex. More recently Siemens-Pyramid and Sun Microsystems have developed and are continuing to develop UNIX cluster-based database server products.

Despite Gray's claim that "NT clusters are easy," perhaps the most telling indicator of the current state of NT cluster technology was the failure to demonstrate 2-node failover! Another surprise: after the big wind-up it was the small stuff that bombed!

Conclusion

I hope my review has given you some sense of my surprise. Be aware that I am *not* anti-NT by any means. On the contrary, from my standpoint NT looks like modern UNIX (commodity Mach) with integrated windows – something I've wanted since my days at Xerox PARC. So, maybe it would be kinder and more accurate to retitle this opening piece: "NT to the Max . . . (NoT) . . . Yet." More, next time.

References

[1] N.J. Gunther, The Practical Performance Analyst, McGraw-Hill, 1997. In press. See <http://members.aol.com/CoDynamo/Book.toc.htm> for publication status.

[2] Anon et al., "A Measure of Transaction Processing Power," *Datamation*, 31(7): 112-118, 1985.

[3] The interested reader can learn more about the TPC-C and TPC-D at <www.tpc.org>.

the Samba file and print server

by Jeremy Allison

Jeremy Allison works at Whistle Communications where he extends and maintains Samba full-time. He has over ten years experience writing UNIX code, and has been working with Windows NT since the first public Win32 Beta.

<jallison@whistle.com>

Samba is an implementation of Microsoft's SMB file and print server protocol implemented on UNIX systems. SMB stands for "Server Message Block," newly renamed CIFS or "Common Internet File System" as part of Microsoft's attempt to make Windows "Internet friendly." Samba allows your UNIX disk and printer resources to appear as share and printer names on a Microsoft network so that users can select them as the icons that appear in the Microsoft "Network Neighborhood" window on their desktop systems.

Samba can also take over much of the functionality of a Windows NT Domain system when serving Windows 95 clients and has many other interesting features.

Samba is software available under the GNU General Public License (GPL) and so can be always be obtained as source code for UNIX systems.

History of Samba

The original author of Samba was Andrew Tridgell, a PhD student at the Australian National University, in Canberra, Australia. In December 1991, with no documentation available for a protocol in use in his laboratory (DEC Pathworks), Andrew examined the packets on the wire and from the packet dumps wrote an implementation of the protocol for his UNIX systems (he called it Server 0.1). The protocol he examined turned out to be the same protocol that Microsoft LanManager systems use. Eventually, after he made his code publicly available, there was enough interest for him to start the "Netbios for UNIX" project (in December of 1993). So he created a mailing list containing all the people who had asked him about his code in the intervening time.

Samba is now a mature product at version 1.9.17 with support from over 100 commercial vendors worldwide. Many hundreds of people have contributed code to the project, which is still being developed at a very rapid pace. The code is now developed by a distributed team of people known as the Samba Team, who work in a manner similar to the Linux kernel developers.

Getting and Compiling Samba

The best way to get Samba is to go to the official Samba Web pages, <http://samba.anu.edu.au/samba>, and look at the "Sources" part of the Web page. The official Web pages are a very good place to bookmark if you will be doing much work with Samba, because announcements of new versions and current issues are made here. Binary versions of Samba are available for some systems (notably Linux) as well, so you may not have to compile Samba yourself.

The latest sources are available as a link to <ftp://samba.anu.edu.au/pub/samba/samba-latest.tar.gz>, which always points to the current released stable version, which at the time of writing is 1.9.17p3.

Download the file into a directory you will use to build the software, and then extract it (it is a tar file compressed with the GNU `gzip` program) by typing

```
gzip -d -c samba-latest.tar.gz | tar xvf -
```

If you don't have `gzip` on your system, you will need to download this useful utility for your system. Try a Web search for a precompiled binary for your system, or to compile gzip up from source code, you can find it in the main source repository for GNU code at : <ftp://prep.au.mit.edu.au/pub/gnu/>

After extracting Samba you will find a directory called samba-<release-version> which for the current version will be samba-1.9.17p3 Change to this this directory and read the documentation.

Samba ships with pages and pages of documentation, and most problems that people have had with it are documented within the docs/ directory. Because we are doing a quick tour here, I'll omit my advice and plough straight into compiling the package.

Samba ships with pages and pages of documentation, and most problems that people have had with it are documented within the docs/ directory.

The main source for Samba is found in a directory called source/ in the tree, and the Makefile within needs to be hand configured for your system. The safest way to do this is to copy the Makefile to a safe place so that you can get back to your original if things don't work. Now edit the Makefile using your favorite text editor and configure it to match your system. Examples for almost all different kinds of UNIX systems, from AIX to SCO, are included. If you are happy to install Samba into the default place (/usr/local/samba), then all you need to do is uncomment the lines containing the FLAGSM and LIBSM definitions for your system (by removing the leading "#" from the line), and you should be ready to compile.

Samba should then just compile without warnings when you type make, and then install itself and all its binaries when you type make install. Note that you should install Samba as root.

There are many different compile options for Samba, all of which are tunable by editing the Makefile. To understand what they all do you need to read the documentation in the docs/ directory. I cover some of the common changes you might make.

Configuring Samba

So you have Samba installed on your system. Now what? I'm assuming you installed Samba into the default place of /usr/local/samba, so the examples will all use this path.

Samba consists of two main daemons and a client access program. The daemons are smbd (for Server Message Block daemon) – this provides the main file service from the UNIX machine – and nmbd (NetBIOS name daemon) – this provides the NetBIOS naming services over TCP that Microsoft networking depends on. You might think they just used DNS like everyone else, but you'd be wrong. The client program, called smbclient, allows an ftp command-line-like access to Microsoft network servers. This can be used to access Windows 95 and Windows NT servers as well as Samba servers. Some people use a variation of this program, smbtar (also provided with Samba) to back up their Windows servers to a UNIX tape drive. All these programs are found in /usr/local/samba/bin (I will assume this directory is on your path for future commands).

Both smbd and nmbd are configured in one file : /usr/local/samba/lib/smb.conf.

It is *vitally* important that this file be writable only by root. Setting it otherwise could compromise your system security.

This contents of this file are based on a Windows ".INI" syntax style, for example :

```
; comments
[section]
  parameter1 = value
  parameter2 = value
```

The file is separated into sections that are surrounded by [] characters, and these are followed by a list of parameter=values pairs, one parameter per line. Comments may be entered on a line starting with a " ; " character.

The main section that controls the behaviors of both smbd and nmbd is the [global] section. This is followed by sections that define how the resources of the UNIX machine are to be exported to the Windows client machines. To examine this in detail I'll show a sample smb.conf file and go through it line by line to explain what each parameter means.

```
[global]
  workgroup = PCGROUP
  security = user
  guest account = pcguest
;
; These next three options set case processing to emulate an
; NT server.
;
  preserve case = yes
  short preserve case = yes
  case sensitive = no
;
; Uncomment the next line and set the correct IP address
; if you wish nmbd to register this UNIX machine with a
; WINS server.
;
;   wins server = 207.76.206.250
;
; Uncomment the next line if you wish this UNIX server to *be*
; a WINS server. Remember there should only be *one* Samba WINS
; server in your network because nmbd does not yet support
; WINS replication.
;
;   wins support = true
;
; Misc options.
;
  mangled map = (*.html *.htm)
  client code page = 850
  veto files = /.AppleDouble/.bin/.AppleDesktop/Network Trash Folder/

[homes]
  guest ok = False
  read only = no
  create mask = 744

[public]
  path = /tmp
  read only = no
  guest ok = True

[printers]
  path = /var/spool/public
  public = yes
  read only = true
  printable = yes
  browsable = no
```

The first three lines in the [global] section

```
workgroup = PCGROUP
security = user
guest account = pcguest
```

are probably the most important. The workgroup = parameter tells nmbd what Windows workgroup the UNIX machine should announce itself as being in. If you have a Windows NT Domain you wish the UNIX machine to appear in, then you should use that name here. This must be the same as the Workgroup/Domain that the Windows machines are in, or they may be unable to see the Samba server in their network neighborhood.

security=user tells smbd that all Windows users connecting to your Samba server must do so with a valid username and password. By default this is sent over the network in clear text, so you should allow this only on local networks known to be safe. Few networks can be categorized as that these days, so Samba can be compiled to support the Microsoft encrypted password negotiation which is based on a "challenge-response" protocol. In fact with Windows NT Service pack 3 Microsoft has disabled the clear text password option on NT by default, although it can still be enabled by changing the registry.

To learn how to compile Samba for encrypted password support, read ENCRYPTION.txt in the Samba docs/ directory. Doing this requires access to a DES (Data Encryption Standard) library, which is not legal to export from the USA at present. This is why Samba does not use encrypted passwords by default. A DES library (in source code form) is available at the Samba ftp site.

guest account = pcguest tells smbd what username to use if a Windows client doesn't present a valid username to the server. By default, such a connection is refused access, but for some services, you may wish to allow it (for public information). The account name used here must exist in the UNIX /etc/passwd file but almost certainly should not be allowed an interactive login (use a new dummy account modelled after accounts such as backup or bin, but use a high uid number).

The next three options :

```
preserve case = yes
short preserve case = yes
case sensitive = no
```

set Samba up to present a case-insensitive but case-preserving view of the filesystem. This is exactly the same as the NTFS or FAT filesystems that Windows NT supports, and many Windows client programs expect such a filesystem. It is possible to set Samba up to be case sensitive or to map all filenames into upper- or lowercase, but the options presented previously are the most common.

For NetBIOS name resolution on Microsoft networks that span TCP/IP subnets, there must be a server known as a WINS (Windows Internet Name Server) server somewhere on the network. This allows Microsoft networking clients that originally only used broadcasts to map between NetBIOS names to IP addresses to find remote servers. Samba has the capability to act as a WINS server and can also register itself with a Microsoft WINS server. The next few commented out lines allow you to choose which you require, but you must uncomment only one of the two possible options. If you have a small, unsubnetted network, you can leave them both commented out.

The options commented as ; Misc options show some of the flexibility of Samba configuration.

```
  mangled map = (*.html *.htm)
```

shows how UNIX filenames can be translated on the fly by Samba for Windows clients (to map `.html` files to `.htm` for standards-challenged Microsoft HTML programs).

```
  client code page = 850
```

is an internationalization option allowing the administrator to define the current code page of the Windows clients. This allows filenames containing the high bit set to be mapped correctly in case-insensitivity cases and to be correctly converted in the UNIX filesystem.

```
  veto files = /.AppleDouble/.bin/.AppleDesktop/Network Trash Folder/
```

is a list of filenames, separated by "/" characters, that Windows clients will never be allowed to see. The given example is how you would prevent Windows clients from seeing the Macintosh metafiles used by the Netatalk AppleTalk fileserver program, afpd, popular at many UNIX sites containing Mac clients.

The following two sections, `[homes]` and `[public]`, are examples of disk resource shares. These are similar to lines in an `/etc/exports` file for NFS sharing, although they can be considerably more complicated. I'll consider the `[homes]` share first.

The `[homes]` share is a "magic" share. If a Windows client tries to mount (or "net use" in Windows parlance) this share, Samba automatically connects the user to their home directory. This saves a lot of time, in that you don't have to define share points for all your users or make them connect to the directory just above the users' home directories in order to get to their files. Windows clients who connect with an invalid username are denied access to the file system (as the first parameter explicitly denies guest access – this is the default but is placed here to make it clear to the administrator in this example).

The line `read only = no` allows the users to write to the share (by default share points are read only). The final line `create mask = 744` causes all files that are created by `smbd` on behalf of the user to have these octal permissions applied to the permissions of the file as a mask. In other words, files created by the users in this share will have write and execute permissions removed for group and other.

The `[public]` share is an example of a publicly writable directory. The `path=/tmp` parameter tells `smbd` what part of the UNIX file system to use for this share, as previously, the `read only = no` parameter allows users to write into this directory, and the `guest ok = True` parameter tells `smbd` to map unknown users into the `pcguest` user mentioned above in the `[global]` section.

The `[printers]` section specifies how Windows clients access UNIX printers (defined in `/etc/printcap`, but System V printing is also supported). This section is a "magic" section like `[homes]`. When a Windows client connects to a printer, the name is looked up in the `/etc/printcap` (or equivalent for System V), and the Windows client is connected to that printer. The first parameter tells Samba where to spool the printer files. The following two parameters (`public`, `read only`) we already know, but the last two are new. The `printable=yes` just tells `smbd` this is a printer share (thus denying regular file access), and the `browsable=no` tells `smbd` not to list this share when a Windows machine requests a list of available resources on a machine.

All these options (and many, many more) are described in the Samba man pages that are installed into `/usr/local/samba/man` when you install the package. Set your `MANPATH` environment variable to point there, type

```
man smb.conf
```

and away you go (for over 30 pages of options, I'm afraid).

Enough of the explanations; let's start up Samba and access some files!

Starting Samba

Samba services can be started in two ways, either as standalone daemons (the normal case) or via inetd. I'll leave the explanation of using inetd to the Samba documentation and show starting as standalone daemons. Starting Samba is as simple as typing (as root) :

```
/usr/local/bin/smbd -D
/usr/local/bin/nmbd -D
```

Note that `smbd` should be started first. To make Samba available on system boot up, just put those two commands into a script, and run them from your local system startup scripts.

Now go to a Windows client machine in the same workgroup and type (in a DOS command prompt window) :

```
net view \\UNIX_MACHINE_NAME
```

where `UNIX_MACHINE_NAME` is the first component of your machine's DNS name. For example, for `testme.whistle.com` you would type :

```
net view \\TESTME
```

If all is well you should see a list of the shared resources you have defined in the `smb.conf` file. If you get the dreaded

```
System Error 53 has occurred.
The network path was not found.
```

you have some debugging to do. Check the file `DIAGNOSIS.txt` in the Samba `docs/` directory.

Assuming everything went well, your Samba server should show up in your network neighborhood within a minute or two

Assuming everything went well, your Samba server should show up in your network neighborhood within a minute or two.

(if not, check out the `BROWSING.txt` file in the `docs/` directory). Try logging on to a Windows PC with a UNIX username and password (assuming that that user has been correctly set up with that password in an NT Domain if you are using NT Domains at your site); you should be able to use the Windows explorer or network neighborhood to browse right into the shared directories on the UNIX system.

Using the smbclient program

The `smbclient` program is a command line program (shell-like) that allows a UNIX user to navigate an SMB shared filesystem. Currently, only Linux allows SMB shares to be mounted directly into the filesystem via the `smbmount` command. All other UNIXs must use this command line to allow their users to manipulate files on an SMB share.

Using `smbclient` is fairly simple (in 1.9.17). Just type :

```
smbclient //server_name/resource_name
```

You will be prompted for a password (the connection is made as the currently logged in user), type in the correct password for your username on the remote server, and you will be given a :

```
smb: \>
```

prompt (where \ represents the path of the directory tree being shared). For example, *Currently, Samba is being extended and developed in more directions than you can imagine.*

to connect to the `public` share on the Samba server we previously set up, we would type :

```
smbclient //server_name/public
```

Note that `smbclient` uses the DNS name of the server, not the NetBIOS name. So you can use a full Internet domain name to get access to a SMB server over the Internet.

Typing a `?` at the `smbclient` command prompt gives a list of commands, currently :

```
ls          dir         lcd         cd        pwd
get         mget        put         mput      rename
more        mask        del         rm        mkdir
md          rmdir       rd          pq        prompt
recurse     translate   lowercase   print     printmode
queue       qinfo       cancel      stat      quit
q           exit        newer       archive   tar
blocksize   tarmode     setmode     help      ?
```

These are rather similar to ftp commands, with a few additions. As usual, the `smbclient` man page will list all of them in gory detail.

Using these commands, you can manipulate files quite nicely on a Windows NT or Windows 95 server, as well as a Samba server. This is handy for remote administration when you don't have a Windows machine on your desk (as in my office).

Support for Samba

Samba has an active mailing list, <samba@samba.anu.edu.au>. To subscribe, check out the Web page for details. Also the newsgroup comp.protocols.smb has active Samba discussions and support.

In the Samba `docs/` directory there is a file `Support.txt` that lists over 100 commercial companies that provide paid support for Samba. Note that the Samba Team cannot guarantee the level of support provided by any of these providers. You must negotiate with the individual companies as to service and rates.

Conclusion

I hope this lightning tour of using Samba has whet your appetite for more information. Currently, Samba is being extended and developed in more directions than you can imagine. To keep up with the latest developments, check out the Samba Web site. All previous versions are kept there in archive form, and even the CVS (version control system) log of the development is available online, so you get to see the developers' check-in comments.

Also on the Web site is a survey form Samba users can fill out to tell the world (and other Samba users) how they are using the software. Currently, there are over 1,400 survey entries and many of the USA Fortune 100 companies (and many others from all over the world) are listed there. I hope your company will be listed there soon !

one size fits all: stretching NetBIOS to fit the enterprise

by Christopher R. Hertel

(with thanks to Dan Shearer and the rest of the Samba Team) Chris Hertel is a network design engineer at the University of Minnesota. He has recently been added to Samba Team.

<crh@nts.umn.edu>

At the University of Minnesota the Network Design team of the Networking and Telecommunications Services Department is collaborating with the Samba Team on a set of enhancements to Samba's NetBIOS nameserver. The nameserver is critical to running CIFS over TCP/IP because it translates NetBIOS names into IP addresses so that CIFS clients can locate services on separate LANs.

[*Editor's Note:* The NetBios name service was originally specified in RFC 1001 and 1002. Microsoft's implementation is called WINS for "Windows Internet Name Server."]

Microsoft-compatible networks at the University of Minnesota are typically supported at the departmental level by local LAN administrators. Like small islands in a large sea, these CIFS LANs are cut off from one another. Each may have its own naming standards, security conventions, domain controllers, browse lists, workgroups, etc.

A while back, we realized that the first step toward unifying these CIFS islands was to provide a university-wide NetBIOS name service. Initially, the idea of installing such a service in a large, hostile, Internet-connected network was daunting at best. In addition to security and management concerns, we had doubts about the scalability of NetBIOS and Microsoft-compatible networking in general.

One particular issue was the desire among the LAN administrators and their users that the NetBIOS machine names would match the DNS Domain names. This correlation is assumed by CIFS implementations and is typically not a problem on smaller networks. At the University of Minnesota, however, there are 200 DNS subdomains that contain a few thousand duplicate hostnames (e.g., ethel.xx.umn.edu and ethel.yy.umn.edu). These duplications are not a problem in the hierarchical DNS because the names are in separate subdomains. Unfortunately, NetBIOS was not designed for large networks. NetBIOS nameservers expect all names to be unique, so it appeared that name collisions would be inevitable.

To resolve these issues, we knew that our NetBIOS name service would need capabilities beyond those in existing implementations. That meant writing our own code, which made Samba the logical choice. The software was robust, the installed base was large, the source code was available, and so were the authors! We found working with the Samba team to be easy and productive, and we soon came up with a list of enhancements that would benefit both the university and the Samba user community.

1. **Performance tuning of Samba's name database.** We needed a name database that would be capable of handling a large number of entries and thousands of transactions per minute. To achieve this goal, we wrote a new database module that makes better use of memory and is much faster. The code also allows alternative database modules to be added so that the nameserver can be adapted to meet different needs.
2. **Access filters.** The lack of access filters is a common security hole in NetBIOS nameservers. Without them, anyone on the network can query or add entries to the name list. (In our case, "the network" is the entire Internet.) The access filters are designed

Although our enhancements are designed to meet our own needs, they are general enough to be useful to any Samba administrator.

to allow or deny query or registration access based on IP address, DNS domain, or combinations of these and other criteria.

3. **External references.** An LMhosts file can be used to prime the name list with static entries. This is useful, but it means creating and maintaining a file that must be loaded every time the NetBIOS nameserver starts. We will supplement the LMhosts file by allowing Samba to use external sources as references. These might include databases, directory services, the DNS, or even other NetBIOS nameservers.
4. **Replication.** The university is made up of several campuses located across the state. We plan to deploy NetBIOS nameservers at strategic locations and replicate the names that are collected by each server. In this way, we will maintain a single set of NetBIOS names across the entire university. Client systems will be able to access the whole list from their local NetBIOS nameserver. Database replication will reduce traffic across WAN links, reduce delays, and provide database redundancy in the case of a network outage.

These features will give us the tools that we need for the creation of a university-wide service. For example, using access filters, external references, and a strict naming convention, we will be able to prevent name collisions in our NetBIOS namespace. Although our enhancements are designed to meet our own needs, they are general enough to be useful to any Samba administrator. Because we are working directly with the Samba Team, our code will be reviewed and merged into Samba, will continue to be enhanced, and will receive the same high-level support. There is a plan to get the database improvements into the next full release (1.9.18). Filters would be next, but have not been completed yet.

telnetd and inetd on Windows NT with OpenNT

by Rodney Ruddock

Rodney Ruddock is a developer of OpenNT at Softway Systems.Several years in the UNIX and POSIX/XPG realm have pulled him deeper into the kernel far away from his degree in Human-Computer Interaction.

<rodney@softway.com>

and Stephen Walli

Stephen Walli is vice-president of R&D at Softway Systems and author of *Go Solo: How to Implement and Go Solo with the Single UNIX Specification* (1995). He has been USENIX's Standards Report Editor and Institutional Rep to the IEEE POSIX committees.

<stephe@opennt.com>

Currently, businesses are faced with the decision to deploy Windows NT. This new operating system promises a robust environment to solve many business application problems, as well as provides a platform to run the many Win32-based packages they count on, all on a price-competitive platform. Organizations are then faced with a set of problems:

- provide tools to integrate and manage the heterogeneous environment
- protect the application development investment in UNIX applications
- ease the transition for UNIX developers and administrators such that the investment in people isn't overstrained

This article takes a look at one approach to the overall problem, briefly presenting the technology (OpenNT) and its application to providing telnetd and inetd.

OpenNT

In September 1995, Softway Systems Inc. entered into a long-term agreement with Microsoft to extend the Microsoft POSIX environment subsystem into a complete traditional UNIX systems environment. The Windows NT operating system is structured in a microkernel-like design with subsystems layered on top of the kernel. Some subsystems are functional subsystems (e.g., the security subsystem); others provide programming and user interface environments. The Win32 user interface and programming environment are provided through the Win32 subsystem. Softway System's technology is provided under the name OpenNT. The OpenNT environment subsystem replaces the less functional, restricted Microsoft POSIX subsystem and provides the user interface and programming environment found on traditional UNIX systems.

With the release of OpenNT 2.0 in May 1996, the OpenNT subsystem provides the following functionality:

- POSIX.1, including the general terminal interfaces
- ISO C standard library
- Berkeley sockets
- System V IPC (shared memory, semaphores, message queues)
- Memory-mapped files
- System V and Berkeley signal interfaces mapped onto the POSIX.1 signal semantics
- Traditional curses (ncurses)
- X11R5 clients (including twm) and client libraries (Xlib, Xt, Xaw, etc.)
- X11R6 server
- Pseudoterminals
- OSF/Motif 1.2.4 and mwm

The architecture of OpenNT allows it to integrate cleanly with the Win32 subsystem.

- cron service
- Tape drive support
- Perl 5 (patchlevel 3)
- The public domain Korn Shell with full job control, the Tenex csh, and 200+ utilities.
- A full telnet daemon that can be run as a Windows NT service, and controlled either through the Windows service manager applet or the OpenNT command-line service program.

The OpenNT subsystem is structured to use the same high-speed facilities for communications between subsystems and the kernel and processes and their subsystems as the Win32 subsystem, so there is no appreciable performance penalty. OpenNT has been designed to ensure it does not in any way compromise the integrity of Windows NT (e.g., security).

The architecture of OpenNT allows it to integrate cleanly with the Win32 subsystem. The two subsystems integrate through the desktop, filesystems, IPC mechanisms such as pipes and sockets, and the ability of OpenNT processes to exec Win32 processes (allowing easy access to Windows NT specific functions via shell-scripted interfaces).

One of the early daemons requested by customers was a "better" telnet service than those available on the market. This drove the integration of pseudoterminals into the OpenNT subsystem. The telnet "service" had to work cleanly as a Windows NT service. Once this work was finished, the next question was "why not inetd?" The next sections describe the effort to bring traditional UNIX daemons over to Windows NT using OpenNT, running them as OpenNT processes, and managing them as services in the Windows NT world.

Porting telnetd to OpenNT

Telnetd is a system daemon that allows a user from a remote location to connect to the system. Telnetd does this in a couple of steps. First it handles the authentication of the user being allowed to login to the system. While doing this, telnetd does a certain amount of perturbation against users who are attempting to break into the system with an iterative attempt password-guessing algorithm. Once the user is authenticated, a shell is started with a controlling terminal for the user. Finally, telnetd provides the communications pathway for the remote user to communicate with the shell. It continues to provide this step until the user's session ends.

OpenNT derives its telnetd from the 4.4BSD source code. The code has seen a number of years of actual use and is very stable and complete in its functionality. The port of the telnetd code began while pseudoterminal support was being developed in the system. Pseudoterminals, or ptys, are a requirement for telnetd to work with the correct semantics.

It was known a priori that certain APIs called from the telnetd source code were not going to be immediately supported for OpenNT 2.0 and were commented out (specifically, the `syslog` functions, which will be available at the end of November 1997). It was also known that the NT security model is quite different from the traditional UNIX model. Specifically, user authentication does not occur against `/etc/passwd`, and `setuid()` does not have the same power as on traditional UNIX systems.

On traditional UNIX systems, "root" is a special user with the authority to do anything. But the same intent can be accomplished by other means while maintaining a tighter security model. With OpenNT, this is done with an extended set of `exec()` APIs named exec_asuser(). This does not cause a change in the telnetd code; rather it causes a change in the method of the user obtaining a shell. (It did cause a simple change in the login source code.)

The exec_asuser() interfaces accept the same arguments as traditional exec() interfaces with an additional argument for user and password. This allows for the exec() of a program with the security profile of the named user, provided that the password is valid. If the security of the original process becomes compromised (refer to CERT notices for examples), the perturber gains only the security profile that the process has – not the security profile of a user that allows anything to be done (i.e., "root"). With OpenNT the process performing the exec_asuser() is usually restricted from obtaining key information about any user. Such data are needed to correctly set up the user's environment. To allow for the correct environment construction, OpenNT has a utility named loginenv(1).

Instead of exec()'ing the user's shell directly, loginenv is started with arguments that include the user's shell. Because loginenv will be running with the security profile of the intended user, it is allowed access to the information about the user. loginenv, after setting up the environment, performs the exec() that will start the user's shell. The shell then inherits the correct environment information from loginenv.

The testing of the port of telnetd was as much, if not more, of a validation of correct pty semantics and behavior as it was of correct telnetd functionality. Once the full support of ptys had been finished, the additional code changes outside of syslog interfaces were few. The changes centered around included header files, how telnetd itself is started, and the addition of an NT-specific login message. The header files changes were <termio.h> to <termios.h> (the POSIX standard).

The change for how telnetd itself starts is not OpenNT specific. The changes made are just as functional on any other traditional UNIX system.

The change for how telnetd itself starts is not OpenNT specific. The changes made are just as functional on any other traditional UNIX system. Historically, telnetd is started only from inetd. A debugging mode is available when started from the command line, but this limits the functionality of telnetd. telnetd was changed so that it can be started from a shell command line with full functionality. Essentially, the debug mode code was extended. This resulted in the addition of some child management code that would normally be handled by inetd. The change also allows telnetd to be added to the Windows NT services control panel via the Windows interface or through an OpenNT utility named-service. With this ability, telnetd can be started automatically when the system is rebooted. This is particularly useful for system administrators needing to interact with machines from a remote location.

The final change was the addition of an information message to explain how NT domains and usernames can be specified together when logging on to a machine. This allows user authentication against the Windows NT domain model. We felt the message was necessary because the Windows NT login authentication was different from what a traditional UNIX user is accustomed to. The alterations were restricted to this so that users employing automatic login scripts (like expect) would not have to make changes.

The Porting of inetd to OpenNT

Like telnetd, the source code base for inetd is 4.4BSD for OpenNT. Currently, inetd is running internally at the OpenNT lab and is expected to be released in the next update (expected in the fourth quarter of 1997).

Porting applications and tools developed on traditional UNIX systems to Windows NT with the OpenNT runtime environment is straightforward.

The changes needed for the porting of inetd can be divided into three parts: updating some of the code to standard POSIX interfaces, removing code not needed (because it is also not supported), and OpenNT-specific issues.

The updating of portions of the code to the POSIX standard centered on the changing of a `wait3()` API call to a `waitpid()` and changing `main` to take two arguments (`argc`, `argv`) instead of three (`argc`, `argv`, `argp`). These minor changes took literally a couple of minutes and improve the overall portability of the source code.

Inetd traditionally reads the `inetd.conf` file to determine which user the child daemon process should be run as. However, as explained earlier, there is not an "all capable" user (root) that can change identity at will. So this class of code was commented out. In the vast majority of cases, inetd does not need a root-level ability. Running inetd as a user with a restricted security profile has its security advantages, too. So the functionality of inetd and its child processes has not been unduly affected.

Changes that are OpenNT specific relate entirely to pathnames. For users accustomed to moving from one system to another, this is not surprising. In the `inetd.conf` file the paths to the utilities that are related to the socket ports that inetd monitors had to be changed. The pathnames still appear as on a single-rooted filesystem. However, all installations of OpenNT do not have the same root.

Windows NT supports the same disk model with letters applied to volumes as DOS and Windows PCs. Often users and Windows NT administrators will want to install certain packages on certain volumes. So the OpenNT world is rooted at `$OPENNT_ROOT`, and some source code expecting there to be a `/bin` directory with particular applications is "surprised." (A future release of OpenNT will provide full single-rooted filesystem semantics.) To accommodate a configurable install root, two interfaces are used to turn a single-rooted pathname to the installed root pathname: `_prefixInstallPath()` and `_getInstallPath()`. The `_getInstallPath()` function was used to change the one occurrence of `chdir("/")` to `chdir(_getInstallPath())`. The `_prefixInstallPath()` function was used in two places in total: (1) for the location of the `inetd.conf` file and (2) for rooting the path of utilities named in the `inetd.conf` file.

Inetd can be managed via the Windows NT service manager application or the OpenNT utility as a Windows NT service. Inetd can be set to run as an automatic service at system boot time, just as with telnetd (i.e., inetd will be started immediately when the system has been rebooted). This has the same advantages for system administrators as mentioned earlier.

In the OpenNT lab, the `inetd.conf` is running with a number of daemons (e.g., `ftpd`, `ntalkd`, and `telnetd`). And the built-in services on well-known ports are also fully functional (e.g., time on port 13). The porting of `ntalkd` consisted of pathname changes in a more minor sense than with inetd and the use of `utmpx` (currently active in the OpenNT lab) instead of `utmp`. The number of lines of change is fewer than ten – most of those dealing with `utmp` to `utmpx` changes (as OpenNT follows the XPG standard).

Summary

Porting applications and tools developed on traditional UNIX systems to Windows NT with the OpenNT runtime environment is straightforward. This includes more complex applications like inetd and telnetd that can be run as Windows NT services and require integration into the Windows NT security and authentication model.

For more information about the OpenNT architecture, please see the paper "OpenNT: UNIX Application Portability to Windows NT via an Alternative Environment Subsystem," presented at the USENIX Windows NT Workshop, August 1997, Seattle, and available at<http://www.usenix.org/publications/library/proceedings/usenix-nt97/walli.html>.

Complete up-to-date information about OpenNT can be found at <http://www.OpenNT.com>.

Win32 Tcl/Tk GUIs on UNIX apps on Windows NT

by Stephen Walli

<stephe@opennt.com>

I hope that this article will expand the boundaries of discussion with respect to how UNIX systems and Windows NT coexist. At Softway Systems Inc., we develop OpenNT, an environment subsystem for Windows NT that provides the programming and runtime environment found on UNIX systems for Windows NT with complete UNIX semantics.

The original Microsoft POSIX subsystem demonstrated the concept of alternative environment subsystems running concurrently on Windows NT. But that subsystem was restricted in functionality to POSIX.1 and ANSI C, and there was little developmental support. The OpenNT subsystem goes beyond the POSIX subsystem and is well integrated with the Win32 subsystem, providing

- integration through the filesystem
- pipes and Berkeley sockets
- the Win32 desktop
- the ability of an OpenNT subsystem process to exec a Win32 app

This sort of integration is used all over the product itself. Once we could exec a Win32 binary, the `lp` command became a simple shell script wrapping `PRINT.EXE` that could reach any printer on the network, rather than needing to build a lot of specialized printer driver functionality. `useradd` and `userdel` were written as simple shell script wrappers around `NET.EXE` to add hundreds of users in batch command line mode. When we needed to write a tool to manipulate the tape drives because some tapes from some UNIX vendors don't play well in this space), we wrote a simple Win32-based `mt` tool using Win32 interfaces that is run from the OpenNT `ksh` command line prior to running `tar` or `cpio`.

One cannot create a single application program that mixes UNIX system calls and Win32 system calls in the same executable.

There is one thing, however, that one cannot do with OpenNT today in the Windows NT world. One cannot create a single application program that mixes UNIX system calls and Win32 system calls in the same executable. Essentially, one cannot have an application calling more than one subsystem as its client. What would the process tree look like if one used `fork()` and `CreateProcess()` in the same application, or what happens when one tries to use both UNIX signals and Win32 exceptions?

There are two places this causes grief today. There is a set of third-party libraries for database access (Oracle and Sybase are the typical requests) that, although provided on Windows NT as Win32 DLLs, are not useful to an OpenNT process as porting tools for the application code being ported from a UNIX system.

The other request typically made is a customer wants to somehow put a Win32 GUI on the UNIX application being ported to Windows NT. So we had a contest in-house to see how many ways one could do this without breaking the portability of the UNIX application source by embedding Win32 GUI calls within it.

Several examples were forthcoming:

- One entry used Visual Basic to rapidly build a forms frontend that communicated with a traditional UNIX server app via Winsock (Berkeley sockets in OpenNT communicating with Win32 Winsock).
- Another entry "did it the hard way" with Visual C/C++ communicating with Winsock to traditional UNIX server code in the OpenNT subsystem.
- There were two entries that used the Win32 Tcl with the Win32 Tk widget set to wrap apps (both traditional and written for the contest) with Win32 GUIs.

The first two entries are interesting because the Win32 client ends can easily be remoted to Win95 machines, but the Tcl/Tk entries started to push the envelope of what can be done on Windows NT while protecting the long-term investment in the entire application.

One entry was a trivial Tcl wrapper around `ps` that displayed the running OpenNT processes one per line and, when a process item was double-clicked, invoked `kill` appropriately to kill the process. The `ps` and `kill` commands are running as OpenNT processes invoked from a Win32 Tcl program displaying with the Win32 Tk widgets, so everything has a Win32 look and feel.

Now comes some of the weird part. The OpenNT X11 server is a Win32-based X11R6 server that displays individual X11 clients in individual windows on the Win95 desktop. These windows are decorated with Win32 decorations for window control, but the widgets within the application are controlled via the window manager (typically via the default one in the server). OpenNT X11 clients (that are OpenNT subsystem clients), both local and remote, can run with the OpenNT server as one would expect, as well as other X11 clients running on other non-Win32 platforms.

The X11 application appears and behaves exactly as one would expect. The OpenNT X11 server allows other window managers to be run, and these window managers run as OpenNT subsystem clients. Currently, `mwm` and `twm` are shipped with different products in the OpenNT family. So by running another window manager, one gets a different look and feel of the X11 client app on the desktop. (The OpenNT X11 server also supports controlling the root window menu from another window manager, so one can use one's `.mwmrc` to pop up a Motif root menu on a Win95 desktop, then run MS Word from the menu.)

One of the early things ported to Windows NT with OpenNT was Tcl and the Tk widget set. So as well as running the Tcl app with a Win32-based wish80 (from a desktop icon or OpenNT `ksh`), using Win32 Tk widgets, wrapping tools like `ps` and `kill` running on the OpenNT subsystem, we could now run the exact same app using the OpenNT Tcl, with the X11-based Tk widget set to give the app a proper X11 look and feel all on the same desktop.

It's a little confusing. But that's actually the point. There are no real boundaries here. One can define and control the look and feel of one's graphic applications to be exactly what one needs and expects. And portability between Windows NT and traditional UNIX systems can take on new meanings with Tcl/Tk that completely protects the entire application's source code investment.

The best entry in our internal contest was a graphic file browser wrapped around `find`, `lc`, and `ls`, with a small "UNIX" tool written to support certain actions. It can be found on the OpenNT tool warehouse on the OpenNT Web site. I am not a GUI programmer (which was why I chose Tcl/Tk to quickly start building graphic apps on Win32), but I would be very interested in finding out about others' experiences in this space with Tcl/Tk on Windows NT.

For more information about Tcl/Tk on Windows NT, please see <http://sunscript.sun.com/>.

For more information on OpenNT for Windows NT, please see "Telnetd and Inetd on Windows NT with OpenNT" in this issue or visit: <http://www.OpenNT.com>.

. . . portability between Windows NT and traditional UNIX systems can take on new meanings with Tcl/Tk that completely protects the entire application's source code investment.

working with Win32: the good, the bad, and the ugly

by David Korn

David Korn is chief software architect in the research division of AT&T Labs. He explores software development techniques that improve programming productivity. His best known effort in this area is the Korn shell, ksh. He is a Bell Labs fellow and an AT&T fellow.

<dgk@research.att.com>

I have been working for over two years trying to write a UNIX interface for Windows NT and Windows 95 based on the Win32 API (Application Programmers Interface) provided by Microsoft.

The result of this effort is a software layer named U/WIN. Information about U/WIN can be found on the internet at <http://www.research.att.com/sw/tools/uwin>. Educational, research, and evaluation copies of U/WIN can be freely downloaded from the net. Commercial versions of U/WIN are available from Global Technologies Ltd., which owns the trademark for U/WIN and licenses U/WIN software from AT&T. The home page for Global Technologies Ltd. is <http://gtlinc.com>.

In this article I describe my experience with the Win32 API, the good, the bad, and the ugly. I had no experience with any previous DOS or Windows system, and my experience with the Win32 API is limited to the subset of functions needed to provide X/Open functionality and does not include graphical user interfaces. The opinions expressed here are mine alone and do not represent those of AT&T or any other entity.

One word of caution about the Win32 API. The Win32 API is not the same on Windows NT and Windows 95. In fact, it differs from one release to the other as new functions are added. More importantly, many of the interfaces are not implemented in Windows 95 or in some cases have different semantics.

The Good

There are a number of aspects to Win32 programming that I found to be quite usable. The association of "handles" with most objects was quite useful. Handles are similar to UNIX file descriptors except that they do not have any ordering property as do file descriptors. In addition to file, each process, thread, and synchronization object is accessed through a handle. Handles can be "duped" across process boundaries because the `DuplicateHandle()` function takes as arguments a handle of both the "from" process and the "to" process.

Handles have associated with them security descriptors that define the owner, group, and access permissions to the object. In addition to the usual read, write, and execute permissions, an object can have synchronization capability. A handle with synchronization capability can be in "ready" or "not ready" state. A process can block until at least 1, or up to 64, handles are in the ready state.

One facility in Win32 that is not available on most UNIX systems is the ability to be notified when files change. The Win32 API has a primitive to create a handle to a directory that has the synchronization capability and will be in place in ready state whenever changes occur to files in that directory or in its subdirectories.

I also liked the thread interface. It was simple to use and provides the necessary functionality without an overabundance of interfaces. One novel feature available only on Windows NT is the ability for one process to create a thread inside another process.

I was impressed with the I/O performance for Windows NT running on a Pentium Pro. It measured up well against Linux with users unlikely to perceive any differences.

I used the Visual C/C++ compiler from command line mode extensively and had relatively few difficulties. The compiler seems to conform with the ANSI-C standard.

Finally, I found some useful sources for information and tools for Win32. There was lots of useful information in Jeffrey Richter's book *Advanced Windows NT* (Microsoft Press, 1993) and several additional useful pointers in Matt Pietrek's *Windows 95 System Programming Secrets* (IDG Books Worldwide, 1995).

The Web site <http://www.ntinternals.com> by Mark Russinovich and Bryce Cogswell is an excellent place to go for useful tools; the set of tools provided by the Microsoft development kit is decidedly lacking.

There were many things that I found bad about the WIN32 API. Foremost among them was its complexity.

The Bad

There were many things that I found bad about the Win32 API. Foremost among them was its complexity. The complex security model is likely to be the cause of security problems because only experts are likely to be able to apply it correctly.

A listing of the exported symbols shows that the kernel32 library has about 675 functions, the advapi32 library, which contains the security interface, has approximately 400 functions, and user32, which provides the user interface, contains approximately 600 functions. The development kit has over 225K lines of included files and comes with over 400 additional dynamically linked libraries.

One example of the complexity is illustrated by the `CreateFile()` interface. It provides the functionality of the UNIX `open()` function. `CreateFile()` takes seven arguments.

1. `pathname`
2. `access` – bitmask with three or more bits
3. `share` – bitmask of read, write, delete
4. `security` – security descriptor + inheritance
5. `mode` – five modes
6. `flags_attr` – 8 flags + 11 attributes
7. `template` – a file handle

There are at least 33,554,432 possible combinations of options for a given pathname, not counting the security descriptor or template file handle.

The `CreateProcess()` function, which replaces the UNIX `fork()` and `exec()` family of functions, takes ten arguments; some of these are structures containing many members. In spite of the rather large number of flag combinations possible, there is no flag setting that gives the functionality of the `exec` family of functions. The `exec` family replaces the current process with a new one. The implementation of the UNIX exec family in `U/WIN` was one of the most challenging tasks.

Another example of complexity can be found in the registry. The registry appears to be an attempt to put configuration parameters in a single place and to provide fast access to them. The registry is structured like a filesystem but has a separate set of API functions to provide access. The number of registry keys is enormous. and their organization is often hard to predict and even differs between Window 95 and Windows NT.

In addition to complexity, a second bad feature of the Win32 API is its incompleteness. Handles can be inherited or duplicated from one process to another, yet there is no API function that given a handle can determine its type. Although the `ReadFile()` and

Other things that I found bad about Windows NT include its scheduling under load, its need to reboot often, and the single-user mentality.

WriteFile() functions can perform asynchronous I/O on a handle (in Windows NT only), the CreateFile() call must have created the handle with the FILE_FLAG_OVERLAPPED flag. Moreover, if the handle was created with this flag, then synchronous I/O on this handle is not possible. In spite of these restrictions, given a handle, there is no API to set, or even to test, what flags have been used to create the file handle.

Another example of incompleteness is the inability to handle the complete filename space. In addition to not being able to create case-distinct directories and to a lesser degree files, the Win32 interface doesn't allow certain filenames such as aux.c and com0.sh to be created. Nor is it possible to access or create files that end in a . (dot). The Win32 API does not provide a call to create a hard link even though the underlying NTFS filesystem supports this.

The third thing that I find bad about the Win32 API is its lack of consistency. This might be classified as "ugly" rather than as "bad" except for the fact that I have found it a source of several errors and much wasted time reading manual pages. UNIX is often inconsistent, and it appears that the Win32 designers didn't learn from mistakes. One area in which the Win32 APIs are inconsistent is in the return values from functions. For example, CreateFile() returns -1 for error, and a handle otherwise. The CreateEvent() function returns 0 for error and a handle otherwise. The RegCreateKey() functions returns the error and you pass it the address of the handle.

The argument usage is also inconsistent. The first argument to OpenProcess() is the desired access. This is also true for the OpenEvent() function. However, OpenProcessToken() passes the desired access as the second argument.

Unlike Windows 95, Windows NT provides a UNICODE interface to the operating system, allowing filenames and other resources managed by the system to be stored in UNICODE. Unfortunately, rather than using variable length UFT8 encoding for the UNICODE characters, characters are stored in 16-bit fixed size packets. The result is that the 7-bit ASCII subset is not compatible at the storage level. In addition, programs need to be compiled either for UNICODE or for ASCII rather than a unified version.

Other things that I found bad about Windows NT include its scheduling under load, its need to reboot often, and the single-user mentality. An example of its single-user mentality is that when you are logged on to Windows NT over a telnet session, ftp prompts for the password on the console, not on the terminal.

The Ugly

In addition to the things that I found bad, there are many things about the Win32 interface that I categorize as "ugly." I think many of the programming conventions and practices used in the Win32 API are poorly chosen.

In the previous section I listed some inconsistencies that I categorized as bad. In addition there are inconsistencies in the API that I categorize as ugly, for example, the inconsistent way functions are named. Many functions such as CreateFile() are named by joining a verb and a noun. I don't care for this convention because related functions tend to be scattered throughout the programming manual. (Yes, hypertext helps!) However, the Win32 API does not use this convention consistently. Functions such as RegCreateKey(), DeviceIoControl(), and DosDateToFileTime() do not obey this convention.

Many of the header files needed to build Win32 applications take a great deal of liberty with the namespace, making it more difficult to program. Although some of these defects exist in the UNIX API, new standards have been more careful in avoiding this problem. The Win32 header files seem to use the namespace in a rather random fashion. Words such as `IN`, `OUT`, `FAR`, `TRUE`, `FALSE`, `DELETE`, `UNALIGNED`, `VOID`, `IGNORE`, `INFINITE`, and `MAXCHAR` are defined by standard headers. Other define constants such as `SP_BAUD`, `GPRT`, `PST_FAX`, `GHND`, and `LPTR`, to name a few, are hard to remember.

Many of the function names in the Win32 interface seem unnecessarily long. This makes programs harder to both read and write. Using long English names provides no positive benefits to non-English-speaking programmers. An example of a long function name is `GetFileInformationByHandle()`. This function could have been named `GetFileInfo()` because there is no other API call to get file information other than a call by handle. As a result of long names and the large number of arguments, function calls often require several lines, making it harder for the eye to spot errors.

I dislike the convention of embedding type information for a variable in its name. For example, the structure instance name `dwProcess` is intended to indicate that the Process element is a double word. This is information that I don't want or need to know. Perhaps a future system will require 64 bits for `Process`. In this case the name would likely change, and I would have to change my code.

I strongly prefer the UNIX conventions of naming types with a `_t` suffix rather than using all caps for type names. By convention, symbolic constants are in all caps, and using them for this purpose makes programs harder for me to read.

I dislike defining at least two and sometimes three types for each abstract type. For example, in addition to the subject identifier type `SID`, there is the type `PSID` which is nothing more than `SID *`. The type `FILETIME` has two additional incantations, `PFILETIME` and `LPFILETIME`.

In summary, after over two years of programming to the Win32 API, I still find it cumbersome and hard to use. I still have to often refer to the manual page. I frequently have to write simple little test programs to find out what a library routine will do under certain circumstances. The UNIX API seems a lot easier, and thanks to U/WIN, I can do all my programming using it and still be able to run the programs on Windows systems.

. . . after over two years of programming to the WIN32 API, I still find it cumbersome and hard to use.

conference reports

This issue's reports focus on the **USENIX Windows NT Workshop**, held in Seattle, WA, on August 11-13, 1997, and on the **Large-Scale System Administration of Windows NT Workshop**, held in Seattle on August 14-16, 1997.

Our thanks to the Summarizers:

George Candea
<candea@mit.edu>

Brian Dewey
<dewey@cs.washington.edu>

Rik Farrow
<rik@spirit.com>

Gus Hartmann
<hartmann@cs.wisc.edu>

Steve Hillman
<hillman@sfu.ca>

USENIX Windows NT Workshop

SEATTLE, WA

August 11-13, 1997

KEYNOTE ADDRESS

Windows NT to the Max – Just How Far Can It Scale Up

Jim Gray, Microsoft Bay Area Research Center

Summary by Brian Dewey

Jim Gray of Microsoft's Bay Area Research Center delivered the first keynote address of the workshop. As one of my colleagues remarked, his talk made him sound like he was from the Bay Area Marketing Center – it was woefully short on technical content. However, Gray was able to put together a persuasive argument that NT is a solid choice for large-scale computing.

Interestingly, Gray conceded to the conventional wisdom that UNIX scales and NT doesn't early in his talk. He even provided his own "incontrovertible" evidence: a graph displaying throughput of the TPC-C benchmark on a variety of platforms over the past 18 months. Using this metric, UNIX servers have not only consistently performed better than NT servers; UNIX servers have been increasing their throughput at a faster rate.

This seems to be pretty damning evidence. However, Gray argued that the performance difference stemmed almost entirely from hardware. UNIX servers run on better hardware than NT. To prove this point, Gray showed the performance of UNIX vs. NT on a six-processor Intel platform. In this chart, NT had comparable absolute performance and scaled better than the UNIX software. Thus, when comparing just the software and not the hardware, NT shines.

Although this argument sounds like it's trying to put the blame on the hardware manufacturers, as much of the blame must go to the Microsoft marketing strategy – NT is targeted for the commodity hardware market. There's a plus side to this: NT is a cheaper solution. Gray pointed this out as well. In a chart as damning to the UNIX advocates as his first one was to Microsoft, he illustrated that an NT solution possesses economy of scale; that is, as you spend more to get higher throughput, your cost per transaction drops. However, the UNIX solutions have a diseconomy of scale. As you kept spending more money to approach the impressive throughput numbers cited in the benchmarks, your cost per transaction would keep increasing. This fact makes NT a more practical solution for most problems.

How large of a handicap is it to NT platforms to not run on the latest, coolest hardware? Not much, Gray argued. The bulk of his talk presented case studies of impressive computational feats performed by NT systems. He warmed up with the standard laundry list: NT has been used to power systems that handle 100 million Web hits per day, 10,000-user TPC-C benchmarks, 50GB of Exchange store, 50,000 POP-3 users sending 1.8 million email messages per day – the list goes on. [*Editor's note:* See Neil Gunther's article on page 9.]

However, the most elaborate example he presented of NT's ability to scale up (i.e., scale to a more powerful, single computer) was the MS Terra Server – Microsoft's proof-of-concept terabyte database. Gray told a delightful tale of how difficult it was to come up with a terabyte of data in the first place, especially data that had to meet the conflicting criteria of being interesting to everybody and not offensive to anybody. Microsoft settled on storing both USGS and Russian satellite imagery in the database. Information from the Expedia world atlas allowed searching. The Web interface to the database was fast and cool; this was an impressive demo with a high "wow" quotient.

Gray also presented a detailed case study of NT's ability to scale out to multiple machines: the billion-transactions-per-day simulation that they developed for the Microsoft Scalability Day show in New York. The BTD demo involved 45 computers (140 CPUs) that ran the TPC-C credit/debit simulation and maintained one billion transactions over the course of one day. Why should we be impressed with one billion transactions? In contrast, Gray told us that Visa handles only 20 million transactions per day.

Although UNIX servers possess a throughput edge, the combination of economics and NT's capabilities make it a formidable platform. However – and this is where things sound suspiciously like an infomercial – that's not all! There's an impressive array of technologies that Microsoft is building in to future versions of NT. Gray briefly listed these: NT 5.0 will have 64-bit memory addressing; Wolfpack provides 2-node failover; a Coda-like filesystem will allow disconnected access to network files; and Hydra provides timesharing on an NT server.

Even though I didn't like the marketing overtones of this talk, I must admit that the reason why most people were at that workshop was precisely that NT is becoming a dominant force in the market. People cannot afford to ignore it. Gray's talk, while not what I'd been hoping for, was probably the perfect introduction to the three-day workshop.

REFEREED PAPER SESSION

Mangling Executables

Summary by Brian Dewey

This session presented four papers dealing with some aspect of binary rewriting on the NT platform. Overall, this was an excellent session; each speaker did a superb job, and the material was interesting. One project, Etch, discussed binary rewriting as a way to analyze and optimize Win32 programs. However, Etch is heavy on the analyze and short on the optimize. Two other papers dealt with optimization in greater depth. Spike is a project from DEC that performs profile-driven optimization of Alpha NT code. Brad Chen of Harvard University presented a paper on Just-in-Time Code Layout; this technique uses a new heuristic for code layout that does not require profiling information. Finally, Anton Chernoff presented Digital's DIGITAL FX!32, a project that allows the execution of x86 binaries on Alpha NT platforms using a combination of emulation and binary rewriting.

Etch

Brad Chen of Harvard University presented Etch, a research project undertaken jointly by Harvard and the University of Washington. It is a tool for instrumenting Win32 binaries, similar to Atom (a tool from DEC that instruments Alpha binaries) and HiProf (a commercial product from TracePoint). By providing a simple interface, Etch tries to hide much of the complexity of the Win32 environment from programmers.

From the programmer's perspective, there are two major steps to using Etch: instrumentation and analysis. During the instrumentation phase, a simple API and series of callbacks give the programmer the opportunity to insert calls to analysis routines at the instruction, basic block, and procedure levels. The analysis routines are the code that will be called during the execution of the instrumented program.

Chen gave a semidetailed example of the steps involved in using Etch to gather an instruction profile. A seven-line instrumentation routine instructed Etch to insert calls to the analysis routine `InstReference()` before every instruction in the application. `InstReference()` was a single-line function that incremented a counter stored in an array that was indexed by the PC. While this is a slightly simplified view of a simple tool, it illustrates the effectiveness of Etch at taming program complexity.

Chen proceeded to give three realtime demonstrations of Etch: instrumented versions of "Maze Lord" and "Monster Truck Madness" that generated a call graph and an animated cache simulation of Notepad. All three demos, while running a tad slow, performed usably well and gathered an impressive amount of data. The call graph tool is similar to the UNIX-based `gprof` utility and provides useful information; the animated cache simulation, although less useful, was certainly flashy and made for a great demo.

One thing that was not mentioned in the talk but is considered in the paper is using Etch as an optimization tool. In addition to instrumenting an application, Etch has the ability to reorder instructions to achieve better performance.

Chen fielded three questions about Etch. Can Etch handle self-modifying code? No, it can't. Etch cannot handle self-modifying code or runtime-generated code, and there are no plans to add that to the project. However, he does not feel this is a serious limitation.

Can Etch handle dynamically loaded DLLs? Yes, it can. Chen explained the Etch team was proud of how well they handled Windows DLLs. Etch will instrument as many DLLs as it can before the program runs based on its static knowledge. However, if the program calls `LoadLibrary()` to use a DLL that Etch did not know about beforehand, Etch will invoke itself on the fly and instrument the new DLL.

Can Etch monitor the kernel? No, it cannot. The Etch team is thinking about adding this capability to a future release; they would like to be able to monitor the entire system.

Spike

While Etch peripherally supports optimization, Spike focuses on optimizing AlphaNT code. Spike is a profile-driven optimizer, which is one of the things that make it interesting. Profile-driven optimization – attempting to make frequently executed code paths faster at the expense of less frequently executed paths – isn't new; it's just tough for the end-user to manage. First, the end-user must instrument the executable and all of the program's DLLs, then run the program to collect the profile information; next, generate an optimized executable and optimized DLL, then remember to run the new version of the program. Spike streamlines this process. Spike's other distinguishing feature is that it optimizes the binary images; there is no need for source code.

To make things easier for the end-user, Spike presents a simple environment for determining which applications should be optimized. Once the user decides to optimize a program – Microsoft Word, for instance – Spike adds it to its database and handles the rest. The user can continue to use Microsoft Word exactly as before.

When Spike sees that a user is attempting to execute a program in its database, it will transparently substitute the instrumented version of the program for the original and gather the profile information. Transparent Application Substitution (TAS) is an interesting bit of sleight-of-hand; because programs may be sensitive to things such as pathnames of the executable, Spike needs to go to some lengths to convince the instrumented program that it is exactly the same application that the user attempted to execute.

Once Spike gathers profile information, it can optimize the executable. This portion of Spike looks just like an optimizing compiler; the input language is Alpha machine code, and Spike builds a standard compiler representation (basic blocks, etc.) of the program. Armed with this representation and the profile information, Spike may reorder basic blocks to keep frequently executed instructions in the icache and minimize branch penalties.

Spike also attempts hot cold optimization (HCO) – along frequently executed (hot) paths, the optimizer may discover that many instructions are executed on behalf of the cold paths. In these cases, Spike removes the instructions from the hot path and inserts compensation code if necessary. Finally, Spike can make improved register allocation decisions based upon its profile information.

The optimized executable is usually bigger than when it started. However, it will be noticeably faster. Their performance numbers showed a 5–25% speedup for the optimized programs, and the real-world applications – as opposed to programs taken from a benchmark suite – showed the larger speedups.

Just-in-Time Code Layout

Brad Chen presented a paper on Just-in-Time Code Layout. Like Spike, it attempts to optimize code layout to improve icache performance. However, it does not require profile information. There are two interesting aspects to the paper: first, the Activation Order (AO) heuristic for code layout, second, the mechanism for dynamic code layout.

Optimizing code layout to improve icache performance is an NP-hard problem. However, compiler writers can use heuristics to improve icache performance. The Pettis and Hansen algorithm attempts to place procedures that frequently call one another close together in memory (e.g., if procedure A() calls procedure B() 10,000 times and procedur› C() 100 times, the Pettis and Hansen algorithm would lay the code for A() and B() closer in memory). The Pettis and Hansen algorithm requires profile information to operate. (Spike uses this algorithm to optimize basic block layout).

Chen's paper proposes an alternate heuristic: Activation Order. As its name suggests, the algorithm arranges code in the order in which the code is called (i.e., the first function called is the first in memory, the second called is the second in memory, etc.). The advantage of this heuristic is that it does not require profile information, and their results show that it performs comparably to Pettis and Hansen. However, this algorithm requires the code layout to occur at runtime. To accomplish this, the link-time generated code segment consists of a series of thunks, one for each procedure.

When a thunk is executed, it performs the following tasks. First, it loads the corresponding code into memory if it isn't already present. Second, it modifies the call site to directly use the dynamically loaded code. Thus, the code copying occurs only once per execution, and the thunk will be activated once per call site.

In addition to improving cache locality, Just-in-Time Code Layout offers another significant advantage: it can reduce overall memory requirements for an application by up to 50%. This is because only the code that is actually executed will be loaded into memory. The Pettis and Hansen algorithm may be able to provide similar advantages, but Chen was unable to quantify this.

JITCL also avoids a problem that plagues Pettis and Hansen when used in the Win32 environment. Because Pettis and Hansen places all frequently executed code at the beginning of a module, it will actually increase cache conflicts between code in different DLLs. DLLs are an inescapable part of the Win32 environment, so the Pettis and Hansen algorithm tended to increase L1 and L2 cache miss rates for Win32-based workloads.

Someone asked if Just-in-Time Code Layout eliminates code sharing among applications. Yes, it does. If two different applications use a DLL that has been engineered to use the Activation Order heuristic, the two applications would share the thunks (which are in the code segment) and not share the dynamically loaded code.

DIGITAL FX!32

Anton Chernoff presented DIGITAL FX!32, a binary translator of x86 NT executables to Alpha/NT executables. This research was motivated by Digital's desire to get a large application base for its Alpha computers and thus had the following design goals:

- Transparency. Users should not need to change the way they interact with their programs. In addition to allowing users to simply click on their favorite Intel binaries, DIGITAL FX!32 needs to handle all of the complexities of the Win32 environment, such as DLLs and OLE objects.
- Performance. X86 code translated by DIGITAL FX!32 needs to perform well. Digital's goal is to get the applications to perform 70% as well as native Alpha applications.
- Correctness. The Alpha code needs to do the same thing as the original x86 code.

The "Transparency Agent" is the component of DIGITAL FX!32 responsible for ensuring that the user never knows that DIGITAL FX!32 is there. It accomplishes this through a technique referred to as "enabling." Any process on the system must be enabled in order to start another process, and enabling consists of inserting hooks to DIGITAL FX!32 code in place of all of the load/execute Win32 APIs. Thus, whenever an enabled process creates a new process, it goes through DIGITAL FX!32 code. The child process will be enabled before it is executed, keeping this process transparent to the user.

The fact that the "enabled" state cascades from parent to child means that DIGITAL FX!32 can enable the entire system if it can enable all top-level processes. DIGITAL FX!32 accomplishes this by enabling the Service Control Manager and RPCSS at boot time.

The only problem is the shell (`explorer.exe`). DIGITAL FX!32 edits the registry to start `fx32strt.exe` as the user's shell; this program starts the Windows Explorer and enables it. All processes that the user will then run will be enabled. However, the Explorer looks at the registry to see if it was the user's shell and behaves differently if it wasn't. To get around this problem, DIGITAL FX!32 needs to temporarily change the registry to let Windows Explorer think it was the shell, then change the registry back so that the DIGITAL FX!32 start program will be run the next time the user logs in. It is, Chernoff admitted, a horrible hack, but it was the only way they could achieve the needed transparency.

The first time a user tries to run an x86 binary, DIGITAL FX!32 uses an emulator. This piece of tuned Alpha assembly gives such good performance that Anton claimed that, on multiple occasions, he would turn off DIGITAL FX!32's translation for debugging purposes, forget to turn it back on, and not notice for several days. The emulator does more than run the executable – it gathers profiling information that is used to drive the optimizer.

The final component of DIGITAL FX!32 is the server, which is implemented as an NT service and orchestrates the binary translation. It manages the profile information gathered by the emulator and is responsible for starting the optimizer. Like Spike, the optimizer is a binary-to-binary compiler, and it can perform the same profile-driven optimizations. The server keeps the resulting Alpha images in a database and may delete translated images to conserve disk space.

DIGITAL FX!32 has several limitations. First, it hasn't met its goal of 70% of native Alpha code, although its performance numbers are still quite good. It cannot handle 16-bit Windows applications; nor does it translate device drivers; users will still need native Alpha drivers for all peripherals attached to their systems. Finally, DIGITAL FX!32 does not support the Win32 Debug API, so a small class of programs (such as x86 development environments) cannot run under DIGITAL FX!32.

There were several questions. Someone asked if DIGITAL FX!32 can handle programs generated by Microsoft's VC++ 5.0? Yes; VC5.0 has not presented a problem.

Another person asked if the registry hack is really necessary. Yes, it is. The developers at Digital attempted many other solutions, but the registry switch is the only one that worked.

Who handles product support for applications transformed by DIGITAL FX!32? Digital does. They view any incompatibility as a bug and will work hard to fix it. However, the very first thing they do on a product support call is retry the steps that generate the bug on an Intel machine. Chernoff said that in at least 50% of the cases, the bug was reproducible in the original program. "DIGITAL FX!32 provides a feature-for-feature and a bug-for-bug translation of the x86 software," he said.

TUTORIAL SESSION

Available Tools – A Guided Tour of the Win32 SDK, Windows NT Resource Kit, VTune, etc.

Summary by Brian Dewey

Compared to UNIX-based operating systems, NT is an infant. One issue this might have for developers is the availability of programming tools. In this session, Intel and Microsoft teamed up to convince the audience that, yes, you can be just as productive using NT.

Intel's presentation was a lengthy sales pitch for the VTune 2.4 CD. The assortment of material does sound impressive.

The backbone of VTune is a tool that passively monitors your system to find code hot spots. When you see peaks in the CPU usage, you can double-click on the chart to get a closer look at what causes the bottlenecks.

Advanced chips and operating systems will offer more opportunities for passive monitoring: VTune can monitor CPU events of the Pentium Pro (cache misses, branch prediction statistics, etc.), and NT 5.0 will notify VTune whenever a module is loaded or unloaded, which increases monitoring accuracy. In addition to passive monitoring, you can use VTune to dynamically simulate sections of code; this will provide a detailed look at processor performance.

The CD also includes "code coaches" that can statically analyze your code for optimization opportunities. This offers two advantages over relying on the compiler's optimizations. First, the coaches are guaranteed to be up-to-date with the latest chip design – something you can't expect from your compiler. Second, the coach can make suggestions for optimizations that the compiler, due to its conservative assumptions, could not implement.

Finally, the CD includes the Intel compilers, the Performance Library Suite, and the Intel chip documentation. Like the code coaches, the compilers will be up-to-date with the latest processor improvements. It offers profile-guided optimization, floating-point optimization, and MMX support. The Performance Library Suite is a set of libraries that are hand-optimized for various Intel chips. Intel will keep these libraries up-to-date with processor improvements.

Louis Kahn of Microsoft told the audience of a large array of tools that will ease the development process. First, he reassured people coming from the UNIX world about the availability of beloved UNIX-based software on NT. Third-party vendors have ported just about every major tool: the GNU suite, X servers and libraries, NFS, etc.

Of course, Microsoft itself makes a large array of development tools for NT. One of the most important is Visual Studio. This bundle includes a string of "Visual"s (Visual C++, Visual Basic, Visual J++, Visual FoxPro, and Visual SourceSafe) that covers most developers' coding needs. Microsoft's Win32 SDK and DDK provide essential documentation and tools. The NT resource kits contain software developed at Microsoft that, for one reason or another, was not included in the release of the operating system. (For example, many of the programs are utilities written by testers that Microsoft decided were useful enough to make available to customers.) The kits are a potpourri of useful tools that will make a developer's life easier. However, Louis stressed that the resource kits did not undergo the stress tests that NT did – they are "use at your own risk."

The Windows Scripting Host (WSH) will allow developers to move beyond the MS-DOS command language – previously the only scripting tool built in to NT. The WSH will execute JavaScript and Visual Basic scripts. Suppliers of other scripting tools, such as Perl or Python, will be able to integrate their products into this architecture. WSH will be integrated with NT 5.0 and is available for download for NT 4.0. Finally, there is the gargantuan Microsoft Developer's Network (MSDN). A universal subscription to the MSDN CDs will buy you nearly every product that rolls out of Redmond – compilers, operating systems, Office & BackOffice, beta software, the SDKs and DDKs, and volumes of documentation. It isn't cheap, but it can be an invaluable resource for serious Windows development. You can find more information for VTune at <http://developer.intel.com/design/perftool/vtune>, for third-party NT tools at <http://www.microsoft.com/ntserver/tools>, and for Windows scripting host at <http://www.microsoft.com/management>.

REFEREED PAPER SESSION

Driver Tricks

Summary by George Candea

This session was chaired by Carl Hauser from Xerox PARC. The audio/visual difficulties challenged the speakers considerably. [*Editor's note:* Some PowerPoint presentations just took off on their own; the speakers would have to stop PowerPoint and recover their place.] The session covered a number of different driver hacks that were used to extend Windows NT.

The first speaker, Bill Carpenter (VenturCom), presented "The RTX Real-Time Subsystem for Windows NT." RTX is intended for kernel-mode tasks that have hard realtime performance requirements; it attempts to solve some of the problems that real-time application developers face when using NT. RTX comes in two forms: extensions to the normal Win32 objects and the Real-Time Subsystem (RTSS).

The talk emphasized RTSS, which tries to make the development of realtime applications easier by taking advantage of NT's rich set of features as well as the availability of off-the-shelf applications. The authors wanted to make the realtime extensions conform as closely as possible to the standard NT interfaces; to this extent, RTX's API supports a subset of the Win32 API as well as additional functionality specific to realtime operations.

Programs that use RTSS are linked with the RTSS library instead of the Win32 libraries and are loaded as NT drivers. However, these processes are not allowed to call NT's driver interface, or they would wreak havoc; they have to limit themselves to using RTX's API. Modifications to the NT Hardware Abstraction Layer (HAL) were necessary in order to allow RTSS to get control of the processor in response to interrupts. As a consequence, RTSS gets to make decisions before anything else in the system. (In Carpenter's words: "RTSS can guarantee response time by stopping NT dead in its tracks.") Three types of objects can be shared between Win32 and RTSS programs: semaphores, mail slots, and shared memory objects. Communication between RTSS and NT is achieved via two unidirectional message queues.

I see some important advantages in using RTX: it offers some of the performance guarantees that realtime tasks require and RTSS maintains most of the neat NT object model, which should make programming easier (note, though, that VenturCom has not implemented resource accounting or protection). It seems that debugging can be difficult at times because RTSS-specific objects cannot be remotely accessed by processes in the other subsystems (e.g., the debugger).

The cohabitation with Win32 processes can also lead to other problems, such as priority inversion (e.g., an RTSS process could be blocked on a mutex held by a Win32 process). The use of a modified HAL as well as having all the realtime tasks running in kernel mode may not be very appealing for nondedicated systems that run other applications besides the realtime ones.

The second speaker was Jørgen Hansen from the University of Copenhagen. His talk, entitled "A Scheduling Scheme for Network Saturated NT Multiprocessors," presented a rich analysis evidencing a problem that occurs in interrupt-driven operating systems: when a device's interrupt rate becomes very high, little or no progress is achieved in the system.

The authors analyzed NT's behavior when network load becomes very high, especially on the receiving end. The network interface ends up generating a high number of interrupts, which results in the CPU spending most of its time running Interrupt Service Routines (ISRs). This leads to the ISRs literally taking over the entire system and significantly affects its performance. As a consequence, user threads are starved because ISRs have a higher absolute priority than any user code.

Furthermore, if the starved threads are the consumers of network data, network buffers can overflow and lead to packets being discarded. This leads to a vicious circle in which CPU time is used for receiving data that subsequently are thrown away; Mogul and Ramakrishnan described this a few years ago as "receive livelock." The problem can become particularly acute when the host has multiple network interfaces

Hansen also described another occurrence, which he termed "thread pinning." This happens in multiprocessors when a given processor may interrupt the network data consumer in order to process the interrupts from a network interface. As a result of the consumer thread not being preempted and the NT scheduling policy, the thread is not assigned to another processor, even though other processors may be idle. This problem can become worse if the consumer thread is part of a multithreaded application and happens to hold a lock on some shared data – the other threads cannot make progress either.

The solution offered by the authors was an extension of Mogul and Ramakrishnan's elegant "polling thread" idea: during high network loads, use a separate thread to poll the network instead of interrupts. They devised a two-layered scheme in which, at low network loads, interrupts are used (in order to maintain low latency), but when the network load exceeds a certain threshold, they switch to a network polling thread (to avoid thread starvation). As a method to decide when thread starvation may occur, they considered three alternatives: monitor the length of network data queues that are emptied by the user thread, monitor the interrupt rate, and measure the percentage of processor time used by the Deferred Procedure Call (DPC) of the device driver. They chose the last one.

The results showed that the two-layer system is effective in maintaining a stable throughput during high network load, and it does not incur a latency penalty during low network load. Systems using locks performed better than before, and there was no more thread pinning. However, the fixed threshold scheme does not perform well when either the network load or the amount of time applications spent per packet varies significantly. The authors also suggested that NT integrate support for both interrupt handling and polling, but that risks transforming a (still) clean system into a hodgepodge.

A member of the audience asked what the contribution of this work was, given that Mogul and Ramakrishnan had already addressed the problem. Hansen said that they had extended the idea to multiprocessor systems (and solved the thread-pinning problem) as well as gone through the experiment of modifying an already existing network driver so that it implements the two-level scheme.

The next paper, "Coordinated Thread Scheduling for Workstation Clusters Under Windows NT," was presented by Matt Buchanan (University of Illinois). He described a way to implement demand-based coscheduling for parallel jobs across the nodes of a cluster of NT machines.

Coordinated scheduling attempts to schedule threads running on different nodes in a cluster in such a way that, whenever they need to communicate with each other, they are both running. This eliminates the context switch and scheduling latencies. Given that today's high-performance networks have latencies only in the tens of microseconds, coscheduling can significantly reduce communication latency. However, coscheduling in clusters is a hard problem because it represents a compromise between the demands of the parallel tasks and the demands of the local interactive processes.

The authors used P.G. Sobalvarro's demand-based coscheduling (DCS) algorithms. The natural way to implement DCS would be to modify the operating system's scheduler, but given the goal of using DCS in clusters running off-the-shelf operating systems (such as NT), this was not feasible. Therefore, they combined a user-level messaging library with a kernel-mode device driver and a hardware-level control program running on the network card. When a message arrives, the control program decides whether to try to preempt the currently running thread and, if yes, issues an interrupt that invokes the device driver. The driver then boosts the priority of the thread that it wants to schedule, hoping that it will get scheduled. Fairness of the CPU scheduling is critical, and it is ensured by a user-level fairness monitor.

One of the lessons learned was that it is very difficult to dictate scheduling policy to the NT scheduler mainly because, in Buchanan's words, "NT scheduling is very well sealed inside the kernel." The lack of appropriate interfaces forced them to use a hack instead of a clean implementation. Unfortunately, the performance results shown were not very convincing, and Buchanan acknowledged that they need to conduct a broader array of experiments.

One of the questions from the audience addressed the problem of scalability, and Buchanan gave a brief "future work" answer. Another participant asked what happens in the case of threads that do not get scheduled as expected. The answer was that this may generate some overhead, but the threads would synchronize later and achieve coscheduling.

The last speaker in the session was Galen Hunt, currently with Microsoft Research. The work he presented, "Creating User-Mode Device Drivers with a Proxy," described a mechanism that allows user-mode drivers to act as kernel-mode drivers under certain conditions.

The architecture of his system consists of a kernel-mode proxy driver and a user-level proxy system. For each user-mode driver, the proxy driver sets up a stub entry and a host entry. The path of an I/O Request Packet (IRP) is from the application through the NT executive, to the stub entry, then to the proxy driver, to the host entry, into the NT executive again, to the proxy service and then to the user-mode driver.

Hunt presented a number of sample drivers: a virtual memory disk (similar to a RAM disk), an Echo filesystem (not related to DEC SRC's Echo) that "mirrors" another filesystem through the driver, and an FTP filesystem that allows mounting of remote FTP servers as local filesystems. The measured performance was worse than for kernel-mode drivers, but that was no surprise, because the user-mode drivers require twice the number of user/kernel boundary crossings that kernel-mode drivers require. Hunt is currently involved in developing a toolkit that would allow the creation of user-mode filesystem drivers. He cautioned though that it would be crazy to replace NTFS with a user-space filesystem, due to performance considerations.

User-mode drivers definitely have some great advantages: they can use all the Win32 libraries, they can be developed and debugged using standard development tools (they can even be written in Visual Basic or Java). Additionally, user-mode drivers can block, be single-threaded, not be reentrant, and not worry about cache or memory management, unlike kernel-mode drivers. These drivers are great for experimenting, and they can be used to emulate nonexistent devices. In addition to that, Hunt expressed the good idea of designing the drivers as Component Object Model (COM) objects that export COM interfaces for the I/O they support. This gives uniformity, and one can even aggregate these components and have drivers inherit functionality from other drivers.

Another advantage is that there were no modifications made to the kernel, which makes it easy to distribute. However, this technique is limited to drivers that do not need kernel-mode access to hardware; the performance of user-mode kernels is comparable to kernel-mode drivers only when physical I/O latency dominates computation and exceeds kernel crossing time.

PANEL SESSION

Do You Need Source ?

Summary by George Candea

Discussing the need for access to the NT source code was definitely one of the hottest sessions of the conference. The panel was organized by Thorsten von Eicken (Cornell) and consisted of Brian Bershad (Washington), Geoff Lowney (DEC), Todd Needham (Microsoft's evangelist), Margo Seltzer (Harvard), Nick Vasilatos (VenturCom), and Werner Vogels (Cornell). It was definitely useful and entertaining. It ended just before Bershad and Needham would have started to throw punches at each other. The motto of the panel seems to have been "If you don't have source, how can you do any work? If you have source, how do you figure your way around it?"

Todd Needham started the discussion by pointing out that Microsoft is interested in providing good source licensing in order to promote research, but it also wants to keep its competitive advantage and continue making money. Currently, Microsoft has 37 research-related source licensees (universities and national labs), of which half a dozen use the source code simply for reference. There is also a commercial license available.

It turns out that if one develops something while holding a research license and then wants to include it in a product, one needs a commercial license. Currently, there is no need for students to sign a non-disclosure agreement (NDA) as long as the university has an agreement in place with the students regarding intellectual property rights. The licensee retains all intellectual rights, but Microsoft gets the license for any software that uses the source code. The license is really meant only for research. One can exchange source code exclusively with institutions that hold a source code license.

Needham mentioned that there are really two ways to get source code: either by having the full source or by using the Software/Driver Development Kits (SDK/DDK) combination. It seems that one other high-demand issue is how to write and install a filesystem in NT. Needham also said that the documentation for NT source is the source itself. (Having hacked for a summer on the NT kernel, I can attest to the fact that the NT kernel and executive code are pretty well documented.)

Seltzer was the next speaker, and she set out to answer the following three questions:

Why do we need source?

Why Microsoft should want us to have source?

What are the obstacles?

In answering the first question, she pointed out that, although NT makes it easy to collect a lot of data, it is sometimes difficult to correlate results collected from different sources. In addition, one major issue in operating system research consists of making hypotheses based on measurements and then verifying them based on the source (typical UNIX approach). So, because of the lack of NT source code, we see that most of the papers reporting research done under NT tend to be wishy-washy when it comes to proving their theories.

The second question's answer is simple: universities are a source of great ideas, interesting code, and bug fixes. Also, students using NT source code will be well versed once they graduate and can make tremendous contributions to the vendors. Besides, research results are directly relevant to vendors.

Finally, a big problem is that there is a large bureaucracy both in universities and at Microsoft. Microsoft requires the university not the principal investigator, to sign the license (as with other source code licenses), and that is hard to do. The university should not be liable for students stealing source code. Also, there are Export Act constraints (e.g., foreign students working on projects), which is why, for instance, Harvard University does not do any top-secret work for the government. One question that clearly arises is whether an approval from Microsoft will be needed before submitting a paper describing work done under NT. That would be very bad.

Lowney was next. Because of some miscommunication, he did not know exactly what the panel was supposed to be about, so he talked about how to do work without having source code. He spoke about Spike, PatchWrx, DCPI, and Atom. At the end, he mentioned that most researchers make modifications that are small and in familiar places, so one possibility would be for Microsoft to provide a "research build" of NT. In addition to that, making executable instrumentation very simple and defining interfaces to interesting modules would be salutary.

Vogels brought a healthy dose of humor to the panel. He used handwritten slides; and, after describing the projects he worked on, he concluded that he would not need source code for any out of his nine projects. He pointed out that the DDK is really for hardware vendors, and the examples are useless for researchers.

Vogels cautioned that NT consists of millions of lines of code and only six pages of documentation, which barely tells you how to get it off the CDs and how to build a kernel from scratch at the top level. You need a 5GB disk (I managed easily to fit it in less than 3GB along with the entire build environment). This amount of code makes it difficult to search for strings; `{findstr /s /I}` doesn't really cut it when you're talking millions of lines of code. The MS Index server, although excellent for Web pages, crashes on source code; and when it works, it comes up with 1,000 irrelevant matches.

The conclusion was that Vogels's group used source code only as documentation (there is no other documentation for NT), examples, and to understand the behavior of NT. It turned out to be useful for debugging, and it led to the discovery of interesting APIs that are not documented or available in Win32.

Vasilatos wanted to convince the audience that VenturCom is not a source code licensee that creates unsuccessful products. [*Editor's note:* Not all of VenturCom's products have been successful, apparently.] It seems they have a license only for the HAL, and they focus on doing embedded systems installations.

Bershad's talk focused on the statement that you don't necessarily need code in order to do cool research (such as Etch and Spin x86). He then addressed the issue of why you might want to have source code: primarily to look at things (documentation, samples, measurements, analysis) and to build stuff (do derivative work that is better, cheaper, and faster as well as new stuff). However, there are reasons why you might not want source: it stifles your creativity and places golden handcuffs around your wrists (Microsoft owns everything). The question is not really whether you need source or not. What you really care about is "do you need information" about the system you're doing research on. And the answer is categorically yes – you need a good debugger, documentation, etc. Simply shipping the source code won't help.

At this point the panel started accepting questions from the audience and a multiperson discussion ensued. Thorsten added that the Microsoft Developer's Network (MSDN) actually has a lot of documentation on the kernel APIs. Someone suggested providing the source code on microfiche, the way VMS did. However, searching it would clearly be a problem. Another member of the audience pointed out that opening the source code would increase people's trust in NT's security. Needham's view on this was that, even though Microsoft doesn't want to achieve security by secrecy, it needs to fix some bugs before releasing the code to universities.

Another important point was that installing NT in a university could be the first large-scale system for which they don't have source code. Having the source proved very useful in the past, and not having it could be a barrier to adoption in organizations. A system administrator said that, when he recommends NT to his customers, he tells them he cannot guarantee that he will solve all their problems. With UNIX, though, it's different: he knows all the internals by heart and can quickly fix the problems that arise.

It turned out to be hard to say whether having source is better than not having it. Historically, we've seen only systems with source. Also, most people who got source licenses do not use source, because the SDK and DDK contain lots of unencumbered source code.

Needham mentioned that Custer's *Inside NT* book is being updated by an outside writer who has access to the source code, and the goal is to make it a *Magic Garden Explained* type of book for NT.

Someone mentioned that there isn't a lot of educational value to having source: having students look at NT in order to learn about operating systems is like having them look at C++ to learn about programming languages. Seltzer responded that NT wouldn't be used in an undergraduate-level OS course, but further on (e.g., graduate level) students gain lots of depth when they look at a real, live OS.

Vogels followed up, pointing out that Linux is very successful among students because they know they have full control over what is on their desktop and that encourages them to experiment with new things (as opposed to using a simulator). Having seen real systems lets students do incremental work without redoing everything.

Seltzer then ran a straw poll in which she presented the scenario of two students, A and B. A had worked only with toy operating systems, whereas B had worked with a real OS. The question was which one would the audience hire. The unanimous answer was B.

At this point, the conversation started heating up. Needham mentioned that in the commercial market, Microsoft seeks a 100% share; but in research, they're not after that.

Bershad said that, in order to have interest in licensing source code, there must be a stake in it. Needham replied that NT still needs to compete. So Bershad asked who does NT really compete with and what happened if Microsoft released the source code. There was no clear answer. Bershad also said that what made Microsoft successful is not the technology as much as its marketing and vision. So what do they lose if they release source? Needham said that Microsoft is new to the university licensing business and the first time they released code was two years ago, so they are still learning and are cautious.

Bershad said the big problem that makes the source license untractable is that, if you do something with NT source, you can't distribute your work. So a suggestion was to model the NT source license after the UNIX licenses. Needham said that the Solaris license doesn't give the freedom Bershad was talking about and that the GNU public license is not something that Microsoft can do. Bershad then said that what we really need is information about the internals.

Seltzer gave an example: if she comes up with a cool filesystem under NT, it becomes Microsoft's property, and it wouldn't let her put it in Solaris. Researchers don't like that. Needham said that was indeed derivative work, and the license would not allow it to be used in commercial products unless a commercial license were purchased. He added that derivative work is what one does as a student (when looking at the NT source) but not what one does based on what one has learned.

KEYNOTE ADDRESS

What a Tangled Mess! Untangling User-Visible Complexity in Window Systems

Summary by Brian Dewey

Rob Short delivered the second keynote address of the workshop. His talk previewed the ways NT 5.0 will decrease the complexity of system management; he told us of the new version's goals and both the obstacles and technologies related to those goals.

One of the most touted features of NT 5.0 is Plug-and-Play – the ability to attach or remove a hardware device to or from your computer and have it work without manual configuration. Short, in this segment of his presentation, merely tried to convince the audience that this was a tough problem. On the surface, this might not seem so bad – after all, a typical computer consists of perhaps two dozen pieces of hardware, and resource allocation is a well-defined and understood problem. In spite of this, 10% of hardware changes will break a user's system.

Two main problems harass Microsoft's Plug-and-Play developers. First is the sheer number of devices. Short told us that NT 5.0 supports nearly 5,000 base system designs, over 4,000 add-in cards, and around 1,200 printers. And these are modest numbers. Short hinted that Windows 95 supports roughly three times as many hardware devices. The immense number of combinations is an inherent obstacle to Plug-and-Play.

Compounding this is the second problem: the lack of hardware standards. When standards do exist, few vendors implement them completely. Short explained this is a by-product of the economics of hardware production – a vendor, faced with the choice of shipping a device that partially implements a spec and taking extra time to redesign the device is almost always better off shipping. Consequently, the burden of making the devices work falls on the software developers.

It's quite a burden; it impairs the ability to implement the most fundamental resource allocation algorithms. For example, when the Plug-and-Play architecture parcels out address space to devices, it needs to deal with cards that have only a limited number of options and others that will alias different memory addresses. Although Plug-and-Play is conceptually straightforward, it seems to be one of the largest obstacles to the completion of NT 5.0.

Even if Plug-and-Play removes the hassles of hardware configuration, users would still be faced with a Byzantine application installation process. In a way, this is the other edge of NT's legacy; you can trace this problem back to the goals of DOS and early versions of Windows. These systems – designed for a single-user, single-computer environment – had an extremely poor separation between user, application, and system resources. NT, while poised to inherit the business of these systems, also inherited the lack of structure. Thus, it seems that every application installs files into the Windows directory, and many of them will have DLLs with the same name. When one version overwrites another, an application that once worked may now be broken.

The solution to this problem is to impose structure on applications. The eventual goal is to have better-behaved applications, and Microsoft has published new application guidelines to help developers. In the meantime, NT will need to force older applications to obey the new rules. To accomplish this, NT 5.0 will lock down the system directory; NT will perform a little directory slight-of-hand to fool applications that insist on writing to system areas. Applications will be distributed in self-contained packages that will facilitate installation and removal, and Microsoft will develop a toolkit that can transform existing, structureless applications into packages.

From a technology standpoint, the most interesting aspect of NT 5.0 is the support for large installations: corporations with thousands of computers on desktops (and increasingly, on employees' laps). Microsoft is attempting to solve two large problems with its large installation technologies. First, existing Windows systems require the end-users (i.e., the thousands of corporation employees) to also be system administrators. Second, existing Windows systems don't provide the tools that the designated administrators need to watch over the thousands of users.

To address these issues, NT 5.0 will provide facilities for automatic installation and updates of the operating system and applications; additionally, NT 5.0 will make it easier to keep the software consistent across the corporation. Administrators can assign policies and applications to groups – this is an essential feature when managing thousands of users. NT 5.0 will also support roaming profiles (users may log on to any machine and get their customized settings) and system replacement (when a computer goes down because of hardware failure, a spare can take its place and get the proper settings automatically).

NT will rely on several new pieces of technology to meet these goals. The first is a Coda-like filesystem. Under this scheme, the master copy of every file will reside on a server; to increase the efficiency for both the client and the server, the client keeps files cached on the local hard disk. The locally cached files also allow users to continue working when their computers are disconnected from the network – say, in case of a network failure or when roaming with a laptop. Upon reconnection to the server, the filesystem will reconcile changes made to local files with the master copies stored on the server.

To increase the practicality of this solution, NT will also include a single-instance store in the server filesystem. If the server detects that two files are identical, it stores only one copy on the server. This will be a "copy-on-write" file – a user who attempts to change a unified file, will get a private copy to modify. The combination of the Coda-like filesystem and the single-instance store will allow nearly the entire `C:\` drive to be a network cache; because the application files will be shared by most users, the single-instance store will minimize the space impact on the server. All applications will be centralized on the server, so administrators will have a much easier job updating applications and ensuring the consistency of applications across the corporation.

The caching and single-instance store also make remote boot an attractive corporate option. By keeping all applications and data stored on a centralized server, a brand-new NT machine can be connected to the network, turned on, and get all of its data automatically. Because the data will be kept on the local disk, this new NT machine will be operable even in the case of network failure, and the common case (i.e., *not* a brand-new computer) will not require extensive data transfers from the remote boot server. Ideally, this technology will allow administrators to plug a machine in, turn it on, and have it work – true zero administration.

Microsoft is doing a lot to make NT systems easier to administer. Some in the audience thought it was odd that Microsoft was adding features to NT in an attempt to combat system complexity. However, the features being added to 5.0 are not merely a marketing wish list; they provide critical capabilities to an operating system that's outgrown the single-user/single-computer environment. The market will judge if this effort works. And given Microsoft's success in the market, I suspect NT 5.0 will be well received.

REFEREED PAPER SESSION

Performance

Summary by Brian Dewey

The papers in this session all dealt with some aspect of performance under NT. Two talked about measuring performance, one about high-performance uncompressed video, and one about adding real-time performance to Windows NT.

Measuring Windows NT – Possibilities and Limitations

Yasuhiro Endo of Harvard University presented this paper; it argued for the development of a new methodology to measure the performance of NT systems. Most benchmarks measure throughput; however, with NT, as with any graphical environment, what users want is quick interactive feedback instead of impressive throughput numbers.

On top of that, the things that tend to infuriate users are the tasks that they expect to be quick but for some reason take a long time. Benchmarks are especially ill suited to diagnose any system with this behavior, because they typically use statistical methods to smooth over any anomalies. Further, the conditions under which today's benchmarks run exhibit little resemblance to the environment in which the computers are commonly used; the test machines are usually disconnected from the network and rebooted before each test to eliminate extraneous factors in the benchmark numbers.

What NT systems need to monitor, Endo argued, is the number of times the system exhibits anomalous behavior that aggravates the user – the instances when the user expects a quick response from the computer yet stares at the hourglass cursor for several seconds or minutes. In addition to merely noting the number of times these situations arise, Endo would like to discover the reason for the slow response time.

To accomplish this, he proposes passively monitoring the entire system. When users experience anomalous behavior, they notify the monitoring tool either through a mouse click on a special icon or – this was my favorite part of the talk – a pressure-sensitive pad that they can punch. Upon receiving this notification from the user, Endo's tool would dump a detailed log of the system's state over the past several seconds, including:

- Per-event latencies. The system will keep a log of how long it takes for every event (mouse click, network packet arriving, etc.) to be processed by the computer.
- Thread status. The system will log which threads are running and which threads are blocked.
- Kernel profile. The tool will closely watch the OS kernel for the duration of this intense monitoring period and write profile information to the performance log.

Using this information, Endo would then attempt to discover the precise cause for the slow response time. However, Endo has not implemented this ambitious plan; the obstacle is the lack of NT source code. Although not necessary for most of the data gathering, Endo argues that the source code is indispensable when attempting to analyze the resulting data. Without source code, he claimed he would be forced to be a "natural scientist": presented with anomalous behavior, he could make a hypothesis as to its cause and devise more tests to confirm that hypothesis. That is the standard operating procedure for a natural scientist; a computer scientist, he believes, would just check the source code for confirmation of a hypothesis. Further, without source, he would never be able to validate his hypotheses, no matter how much he tested them.

Microsoft will license its source code; however, the licensing agreement is unacceptable to Endo and the Harvard lawyers. The primary sticking point is the confidentiality agreement; Endo has understandable reservations about signing something that may prevent him from publishing his results in the future. The lawyers have quibbled over the signature authority; Microsoft wants the university to sign the agreement, but the university believes it is the job of the principal investigator.

I was disappointed by this presentation, but I liked the paper. I thought the design of a new testing methodology was both interesting and useful, and I wish Endo had been able to implement his design and provide us with results. However, the balance of the presentation was very different from the balance of the material in the paper.

In the paper, Endo and Margo Seltzer go in depth into the proposed design and relegate the complaints about the source licensing agreement to a single, short paragraph. However, in the presentation, there was less detail about the methodology and proportionately more time spent discussing the flaws in the way Microsoft does business with the research community. Thus, I was left with the impression that Endo was making more of a political statement than a contribution to the research community. I'm glad to say the paper proved me wrong.

Measuring CIFS Response Time

A second paper picked up on the response time theme: "Adding Response Time Measurement of CIFS File Server Performance to NetBench," presented by Karl Swartz of Network Appliance. The impetus for the paper was the fact that NetBench, the most widely used PC-oriented benchmark, measures only fileserver throughput. (This stands in contrast to SPEC SFS, used to analyze the performance of NFS fileservers, which measures both throughput and response time). The paper described an addition to the NetBench measurements that accounted for response time.

Swartz did not have access to NetBench source code, so it was impossible to modify the benchmark itself. He overcame this problem by putting a packet sniffer on the network. The packet trace was then analyzed offline. By matching client requests with server responses, he was able to compute response time.

Armed with the ability to measure both response time and throughput, Swartz gave some preliminary numbers comparing Network Appliance's F630 fileserver with a Compaq ProLiant 5000 running NT 4.0. (Interestingly, this is the only place that NT directly enters this paper; the NetApp F630 runs proprietary software and doesn't handle the NT extensions to the CIFS protocol.) Not surprisingly, the F630 performed well compared to the Compaq on both the throughput and the response time metrics.

What was surprising was that the Compaq's throughput, after dropping off once the benchmark exceeded 20 clients, started increasing once the working set exceeded the amount of memory on the server. This seeming paradox was explained by the newly obtained response time numbers, which showed a dramatic increase in response time over the same period. Swartz hypothesized that NT switched algorithms and sacrificed response time to improve throughput when the server was under a heavy load.

High-Quality Uncompressed Video over ATM

This was a difficult presentation to follow, in part because the sound system wasn't working well and in part because Sherali Zeadally, the presenter, spoke at an amazingly rapid clip. Luckily, I had the paper to refer to when transcribing my sketchy notes!

This research addresses the issue of sending uncompressed video over a network. Although the paper and the talk briefly touched on all of the major components of a viable system that uses uncompressed video, such as the large amount of disk storage needed (1GB for 45 seconds!) and tools for multimedia editing, the focus was on the network bandwidth requirements.

Zeadally spent a large amount of time in his presentation justifying the need for uncompressed video. His argument rests on two points. First, the delivery of uncompressed video reduces computational overhead. Second, and more important, uncompressed video is artifact-free; this is a crucial benefit for applications such as medical imaging. (Lossless image compression, which is also artifact-free, doesn't gain enough compression to be worth the computational overhead, Zeadally argued.)

The major obstacle to a high-bandwidth application such as uncompressed video has been slow networks and slow workstation busses. However, these obstacles are eroding. Currently, however, a designer of a high-bandwidth application needs to rely on custom or proprietary hardware to deliver the required performance. Zeadally's research attempts to deliver the uncompressed video using an off-the-shelf, open architecture. His prototype system uses DEC alpha workstations running Windows NT 4.0 with a PCI bus and an OC-3 ATM network.

To test the system, Zeadally used video captured at 15 frames/sec; sending this over the network required throughput of 110 Mbits/sec (the ATM network was capable of 155.52 Mbits/sec). He measured the application-application throughput for both TCP/IP and UDP/IP. Both protocols were able to deliver the required throughput. The TCP/IP test kept the CPU utilization at around 55–60%; for UDP/IP, the utilization was around 50%. Zeadally explicitly notes in the paper that these results exonerate Windows NT from the charge that it cannot deliver high performance over an ATM network; he cited other bottlenecks in previous researchers' results that led to their low throughput numbers.

As with Endo's talk, I was much more satisfied with Zeadally's paper than with his presentation. During the talk, I was too busy keeping up with the breakneck speed of delivery to process the contributions that the paper made. This research is an interesting proof of concept for the viability of network applications based on uncompressed video, and it provides an amazing example of what's possible with commodity computer technology. Unfortunately, my ears proved to lack the bandwidth required to process this talk and reach this conclusion in realtime!

Dreams in a Nutshell

Steven Sommer presented Dreams, a set of realtime extensions to NT 3.51. Traditional realtime systems and conventional operating systems work in very different environments. One of the largest jobs of a conventional operating system is to protect processes from one another. A realtime system can assume that all of the processes in the system are cooperating toward a common goal and therefore do not need the types of protection that a conventional OS provides. However, a realtime system needs to ensure that the different processes on the system are able to meet their realtime deadlines. The goal of Dreams is to combine both worlds by adding the capability for "temporal protection" to NT: processes can now specify realtime deadlines, and NT will protect those deadlines from other processes, just as it protects the address space and other resources of processes.

The building block for temporal protection is the "Transient Periodic Process." A TPP has a period, which specifies how frequently the process needs to run, a deadline, which specifies how long the process has after the start of the period to get its job done, and an expected execution time, which is how long it thinks it needs each period to accomplish its task. When an application wishes to create a TPP, it sends a request to the process manager, which in turn talks to the reservation manager.

The reservation manager performs a schedulability test for the TPP. If it passes this test, then the process is accepted. At this point, the operating system guarantees that the process will get its reserved time and that it will miss its deadline only if it uses all of its expected execution time. A TPP is said to "overrun" if it uses all of its expected execution time without completing. The Dreams scheduler may allow an overrun process to continue executing, but only if there is no other nonoverrun TPP that is ready to execute. The Dreams system has a schedule enforcer that will preempt overrun processes if necessary.

Although the realtime extensions are interesting in their own right, the project makes contributions through its implementation that are relevant outside of the realtime community. First, Sommer's team implemented most of Dreams as an NT subsystem; therefore, they needed to make only minor modifications to the NT Executive and the LPC mechanism. Even the Dreams scheduler (which selected the realtime thread to run next) lived in the subsystem. This led to both a clean design and to code that was easier to test and modify than if it had been placed in the kernel.

In this respect, Dreams is a persuasive case study for researchers who want to extend the capabilities of NT in similar ways. The two-tiered scheduling design that resulted from the Dreams scheduler living in the subsystem hid much of the scheduling complexity and made for a simpler model. Further, Dreams needed a system of priority inheritance to ensure that if a realtime thread was waiting for a regular system thread to complete, the system thread would inherit the realtime priority. The priority inheritance improved the performance of the system as a whole, and Sommer argued that it would make a valuable addition to NT even in the absence of realtime needs.

INVITED TALKS AND PANEL

Building Distributed Applications – CORBA and DCOM

Summary by George Candea

This session was moderated by Carl Hewitt (MIT) and consisted of Peter de Jong (HP) and Nat Brown (Microsoft).

Hewitt started by giving an overview of CORBA and COM. He said COM was what Microsoft used and CORBA was what everyone else did. He then pointed out that, due to market pressure, CORBA needs to bridge to COM and vice versa. Thus, CORBA evolves and includes features that COM has and CORBA doesn't (such as unique identifiers for interfaces and unique naming of object factories). The reverse is true as well: there are missing capabilities in COM that exist in CORBA (e.g., unique identifiers for objects, well-systematized runtime information from repositories, and class hierarchies).

Another problem is that certain features (e.g., persistence) are different in CORBA and COM. Hewitt mentioned that both CORBA and COM lack transparency and simplicity. Not only is "transparency heaven in this business," but both are highly complex and used mostly by wizards. This leads to numerous opportunities for errors, exceptions, misunderstandings, and subtleties. Cross-platform development is very challenging.

An interesting point that Hewitt made was that there is a new kid on the block that causes both CORBA and COM to evolve even faster: Java. It has nice things such as garbage collection (within a few years they will even have distributed garbage collection). Java has a lot of metainformation at runtime, which COM doesn't have (cannot inquire what the methods are and what the objects are). Components are also alive in Java before you actually run your compiler on your code. So COM and CORBA have to do Java. No more Interface Definition Languages (IDL)! Everybody hates IDLs, and Java lets you avoid them.

de Jong essentially discussed a "top ten list" of CORBA's advantages: language heterogeneity, components, transports, application coordination, reuse of services, interoperability, interworking, computation tracking, resource tracking, and scalability. He said the heterogeneity of languages on CORBA was excellent, because it supports C, C++, Java, Ada, Smalltalk, and Cobol. (PARC's ILU also supports Common LISP and Python.)

Brown then talked about COM, emphasizing its scalable programming model: in the same process you use fast, direct function calls; on the same machine you use fast, secure IPC; across machines you use the secure, reliable, and flexible DCE-RPC-based DCOM protocol. He then introduced DCOM as COM++. DCOM adds pluggable transports (TCP, UDP, IPX, SPX, HTTP, Msg-O) between client machines and servers and pluggable network-level authentication and encryption (NT4 security, SSL/certificates, NT Kerberos, DCE security), does connection multiplexing (single port per protocol, per server process, regardless of number of objects), is scalable, and uses little bandwidth (header is 28 bytes over DCE-RPC; keep-alive messages are bundled for all connections between machines).

Further areas of research for COM seem to include high-level language integration, application management ease of use, and deep/robust extensibility. Brown chose to mention what he thinks is wrong with CORBA: focus is on cross-node or networks reuse/integration, which is not practical for horizontal reuse/integration; incomplete specification (e.g., marshalling format of certain types of data structures or implications of lack of services such as naming, events, lifetime management); and it is not architected for extensibility.

A number of questions followed. One of the first ones asked was what the real long-term solution to authentication, especially in the context of interoperability. The answer was that this is a hard problem due to object delegation, which poses the question "who is the object talking for?" Brown said that DCOM is just taking a stab at the security and authentication problem using role-based security. Hewitt added that the systems need to include auditing because they are extremely complex.

Someone mentioned that last October Microsoft had said it would give ActiveX technology and specifications to the Open Group, for integration. Brown said the Open Group indeed has all the source code and specs, but it is moving slowly. The person asking the question said The Open Group is claiming that Microsoft is the reason for the delays. The question remained up in the air.

When someone asked for a comparison of the scalability of the two models, neither de Jong nor Brown could make a convincing argument that they scaled well.

Another question asked how much traffic the keep-alive messages generated in DCOM. Brown said that every two minutes there is a 40-byte UDP packet (with security disabled) going between every pair of machines (whenever no logical connections exist). Over TCP the keep-alive traffic would be even higher (they need to fix this).

REFEREED PAPER SESSION

Distributed Systems

Summary by George Candea

The first paper in this session was "Brazos: A Third-Generation DSM System." The motivation for the work presented in this paper was based on the observation that technological advances in networks and CPUs have made networks of workstations a viable replacement for bus-based distributed multiprocessors. A "third-generation" DSM system, which uses relaxed consistency models and multithreading, seems particularly appropriate for networks of multiprocessor PCs.

One of Brazos's core components is an NT service that must be installed on all the Brazos machines. This service is responsible for receiving and authenticating incoming DSM requests. Also, in order to provide a way for system threads to update memory pages without changing the page's protection, Brazos makes use of an altered `mmap()` device driver to mimic the UNIX `mmap()` call.

There are a number of differences between UNIX and NT, and some of the ones relevant to the implementation of a DSM system are NT has native multithreading support but doesn't have signals, NT's use of structured exception handling, and the TCP/IP stack that is implemented through a user-level library (WinSock).

Brazos attempts to take full advantage of NT features, such as using multithreading to allow computation and communication to occur in parallel, using multicast as a means for reducing the amount of data sent over the network, and using scope consistency to alleviate false sharing of pages when threads are actually updating separate data elements. Multicast turns out to work especially well in time-multiplexed networks, such as Ethernet, because the cost of sending a multicast packet is the same as sending a regular packet. Brazos also tailors data management at runtime, as a function of the observed behavior.

One significant advantage of Brazos, and good DSM systems in general, is that the programmer can easily write programs that access memory without regard to its location. However, the programs must be linked with a static Brazos library. Brazos does increase performance, but not in a very dramatic way. The performance slide did not have the axes labelled, but if I interpreted it correctly, the biggest speedup (of 1.64) was obtained on an application that takes advantage of the scope consistency model.

The second paper was "Moving the Ensemble Communication System to NT and Wolfpack," presented by a humorous Werner Vogels (Cornell). Werner debuted his talk asking, "If you were in the emergency room, would you trust NT to run the systems that take care of you? What about if you were on a plane? What about trusting it to drive your NYSE network? Our goal is to make you feel comfortable with these situations." Ensemble attempts to add reliability, fault tolerance, high availability, security, and realtime response to clusters of computers.

The emphasis of the talk was on the issues that came up during the migration of Ensemble from UNIX to NT platforms. For example, the new Ensemble is coded in OCaml (dialect of ML), which allows the authors to use a theorem prover to verify correctness. The OCaml runtime under Win32 had to be extended in order to allow for the different interface semantics of Win32 (e.g., files and sockets are different types of objects under Win32, whereas they belong to the same type in UNIX). But most of the porting time went into developing a common build environment for UNIX and NT. Ensemble source is currently maintained under UNIX, and the necessary Win32 make and dependency files are generated at checkout time.

The NT version of Ensemble has COM interfaces, which allow for increased flexibility. Applications using a standard DCOM interface can get transparent replication from Ensemble. The authors also used Ensemble to strengthen Wolfpack by adding higher availability and scalability (through software management of the quorum resource), support for hot standby (by using state-machine replication techniques), and programming support for cluster-aware applications.

The performance slide, which showed encouraging results for Ensemble, assumed that all the Ensemble servers were local. Werner acknowledged that if the servers were distributed over a WAN, the performance results would be completely different. One of the questions asked what mechanisms made Ensemble cheaper than DCOM. Werner said that the marshalling in DCE-RPC is very generic and expensive, but Ensemble's protocol itself (as well as the marshalling) is much cheaper than DCE-RPC. Also, the way the object resolver on the server side works is less complex than the one distributed by Microsoft.

REFEREED PAPER SESSION

We're Not in Kansas Anymore

Summary by George Candea

Partha Dasgupta from Arizona State University presented the paper entitled "Parallel Processing with Windows NT Networks." He was one of the speakers who believes NT has many features that make it better than UNIX. He described (with fancy, colorful slides and animations) the techniques they used and what they faced when moving their parallel processing systems to Windows NT.

When porting Calypso, they came across some important differences between UNIX and NT, most notably the fact that NT does not support signals; it uses structured exception handling, has native thread support, does not have a remote shell facility, and applications are expected to be integrated with the windowing system. Also, the NT learning curve is very steep for UNIX hackers.

Chime is a shared memory system for Compositional C++ for a network of NT workstations. Chime tries to achieve such goals as structured memory sharing and nested parallelism. Built from the very beginning on NT, Chime took advantage of NT features that, according to Partha, are more elegant than under UNIX: user-level demand paging, support for handling thread contexts, and asynchronous notification. Also, in the context of the other projects, threads turned out to be an elegant solution for process migration, distributed stacks, and in segregating functionality. Some other advantages of NT are the good program development environment and tons of online documentation. The end results obtained with Calypso and Chime under NT were comparable to the results obtained on UNIX.

One of the lessons that Partha suggested to take home was not just to modify applications for NT and recompile; rather, redesign them in an "NT-centric" way, so they can take advantage of the operating system's features. He also cautioned that Microsoft's terminology can be confusing for UNIX people: the "Developer's Network" (MSDN) is not a network, the "Developer's Library" is not a library, the "Resource Kit" contains nothing about resources, and "Remote Access" does not let you execute anything remotely.

The following two papers generated a lot of discussion and were definitely followed with great interest by the audience. The first one, "OPENNT: UNIX Application Portability to Windows NT via an Alternative Environment Subsystem" was presented by Stephe Walli from Softway Systems. He started his talk with Walli's First Law of Application Portability: "every useful application outlives the platform on which it was developed and deployed." Walli emphasized the need for writing applications to a particular model of portability (such as POSIX); porting to new platforms that support that standard would be much easier.

There are a number of ways in which UNIX applications can be ported to NT: complete rewriting, linking with a UNIX emulation library, fiddling with the NT POSIX subsystem, or using the OpenNT subsystem. A number of elementary programs can simply be recompiled under NT, and everything will work fine; but most of them use operating system resources, which make the port much more difficult. Using the POSIX subsystem can sometimes be a source of surprise: major aspects work as expected (e.g., signals), but there are many details that are different from the "UNIX-style" POSIX.

The goals of OpenNT were, among others, to provide a porting and runtime environment for migrating source code from UNIX to NT platforms, to ensure that any changes brought in to the source code will not make it less portable (by being NT specific), and to ensure that NT's security is not compromised. OpenNT is an NT subsystem consisting of the subsystem executable, a terminal session manager, and a dynamic link library. OpenNT currently supports POSIX 1003.1 (including terminals), the ISO C standard library, `mmap`, System V IPC, `cron`, `curses`, Berkeley sockets, System V and Berkeley signals, etc.

Win32 and OpenNT share a common view of the same underlying NT File System (NTFS) (OpenNT does not add any UNIX-ish filesystems), and OpenNT adds some functionality above that. POSIX permissions are mapped from the file's ACLs. There is no `/etc/passwd` or `/etc/group`, and the standard `/usr` and `/bin` do not necessarily exist, which means that applications would have to be redesigned if they assumed their existence. Security and auditing features available in NT are available to OpenNT applications as well. An OpenNT terminal is a Win32 console. Cut-and-paste works flawlessly between Win32 and OpenNT applications. The OpenNT X11R6 server is a Win32 application. There is a telnet daemon that ships with OpenNT and is a direct port of the real telnetd.

Some of the performance results are interesting: CPU-bound applications exhibit the same performance under Win32 and OpenNT, and they run faster under OpenNT than under a traditional UNIX. Disk performance under OpenNT is close to that under UNIX. OpenNT is outperformed by Win32 on small block I/O but does better than Win32 on large blocks. Also, socket throughput seems to be the same, independent of which system is being used.

The next paper, "U/WIN – UNIX for Windows," was presented by David Korn from AT&T Labs. He talked about the UNIX interface layer that he wrote on top of NT and Win95. Korn said there are few if any technical reasons to move from UNIX to NT and that the main motivation for his work was to serve those people who need the large collection of existing Windows software and the more familiar GUI but still want to run their favorite UNIX applications without having to make Win32-specific changes. The result of this was U/WIN – a set of libraries, headers, and utilities.

According to Korn, Microsoft has made the POSIX subsystem useless because there is no way to access any other functionality in addition to the 1990 POSIX 1003.1 standard. OpenNT, which tries to enhance this subsystem, still does not allow mixing of POSIX and Win32 calls.

Korn's U/WIN takes a different approach from OpenNT: he designed a UNIX interface that wraps around Win32. Thus, U/WIN consists of two DLLs (that implement the functions in sections 2 and 3 of the UNIX man pages) and a server process named UMS (which generates security tokens and keeps `/etc/passwd` and `/etc/group` consistent with the registry). UNIX applications can be linked with the libraries, and (if this step works) they can run under NT. U/WIN supports Universal Naming Conversion (UNC), `fork()` and `exec()`, special file names (e.g., `/dev/null`), and absolute file reference (e.g., `/usr/bin`). Signals are supported by having each process run a thread that waits on an event and is woken up whenever it has a signal to read. The `termios` interface is implemented with two threads connected via pipes to the read and write file descriptors of the process. `stat()` and `setuid/setgid` bits are stored using the multiple data streams feature of NTFS. Sockets are implemented on top of WinSock.

Some of the problems encountered while writing U/WIN were NT's filesystem naming (which does not allow certain characters in file names), the presence of special files (e.g., `aux` and `nul`), line delimiters in text files, inconsistent Win32 handle interfaces, inconsistencies in the way Win32 reports errors, etc. Korn also brought up an interesting point: if a UNIX emulation layer is considerably slower than Win32, then programmers will rewrite their applications to use the native Win32 calls, thus rendering the emulation libraries useless in the long run.

There are currently 175 UNIX tools (including `yacc`, `lex`, and `make`) that have been ported to NT using U/WIN. There are also a number of outstanding problems that still need to be worked out (e.g., authentication, concurrency restrictions, and permissions). Performance of U/WIN at this point is not very good. Although there is no loss in I/O performance, `fork()` is about three times slower than on UNIX, `vfork()` is about 30% slower, and file deletes are about two times slower. [*Editor's note:* Articles by Korn and Walli appear in this issue.]

KEYNOTE ADDRESS

Operating System Security Meets the Internet

Butler Lampson,
Microsoft Corporation

Summary by George Candea

This keynote speech focused on defining what security is today, how operating systems achieve security, how networks achieve security, and how the two efforts can be put together.

Lampson pointed out from the very beginning that computer systems are as secure as real-world systems – neither more nor less. This translates into having good enough locks to prevent "the bad guys" from breaking in too often, having a good enough legal system that punishes "bad guys" often enough, and generally witnessing little interference with daily life. Computer users face the normal dangers of vandalism, sabotage, theft (of information, money, etc.), and loss of privacy. These plagues are typically caused by bad programs and people, where "bad" can be either hostile/malicious or buggy/careless.

In spite of the seemingly acute need for security, we still don't have "real" security. This is because the danger, overall, seems small, so people prefer to pay for features rather than for security. In addition, setting up secure systems is complicated and painful.

At the level of the operating system, users assume there is a secure communication channel to/from them. The OS then authenticates users by local passwords, and access to each resource is controlled using, for example, access control lists (ACLs).

The only difference in network security, according to Lampson, is authentication. In distributed systems, security becomes hard because the systems are very big and consist of heterogenous and autonomous parts that interact in complex ways. Such systems are also designed to be fault tolerant, meaning that they could be partly broken but still work properly; this makes authentication hard. Some systems try to circumvent these problems. For example, Web servers simplify things by establishing a secure channel via SSL, thus reducing the problem to that of OS security. Web browsers authenticate servers by SSL in conjunction with certificates (note that DNS lookup is not a secure way to authenticate servers). Browsers also authenticate programs by verifying digital signatures.

Lampson gave an overview of how OS and network security can work together, using the concept of principals – abstractions of people, machines, services, groups, etc. Principals can make statements, and they can speak for other principals. For secure distributed systems, we need to have network principals be on OS ACLs, allow network principals to talk for local principals, and assign secure IDs to network principals. As an example, he briefly described SDSI (Simple Distributed Security Infrastructure) and SPKI (Simple Public Key Infrastructure).

The talk was followed by an avalanche of questions, which clearly indicated the great interest that security generates and the fact that few people really know what security is all about. Lampson's first answers indicated he believes that firewalls are "the right thing." He also pointed out that people running <www.microsoft.com> should worry a lot about denial of service attacks, but not as much about the content, which can be easily regenerated. However, this situation could change in the future. He also expressed frustration with the fact that distributed systems research doesn't get deployed due to security concerns (the Web is not a real distributed system).

Lampson was asked what he thinks about the security of incoming email attachments, and he answered that, essentially, any form of executable code should carry a digital signature that is verified before the code can gain access to your system. In the context of auditing, someone asked what would happen if one got root access by some obscure means and then destroyed its trails. Lampson said that root needs to be fully audited as well, and if the system allows for the audit trails to be tampered with, "you're in soup."

The last set of questions had to do with distributed security systems. A member of the audience asked what the universal solution to revoking certificates would be. The answer was that one will always need to rely on some sort of timeout after which certificates are revoked.

Someone asked whether Lampson thinks worldwide authentication systems would become popular and whether it is the right model. He answered that, as long as the communicating parties agree on an encryption key, the system doesn't necessarily need to be worldwide. Another question concerned the vulnerabilities of such worldwide systems; the answer was (1) getting compromised and (2) it is difficult to get people to agree on things.

The last question referred directly to SDSI and asked how the system could be policed. Lampson said the legal system will work at its own pace (a couple of decades) and provide a framework for this. He also added that reliable auditing across the Internet is not possible.

CASE STUDIES

Deep Ports

Summary by Brian Dewey

A company faced with the challenge of porting an existing UNIX-based application to Windows NT has a choice of two strategies: a shallow port that preserves most of the application's UNIX flavor, or a deep port that involves rewriting key parts of the application to fit into the Windows NT model. This panel discussed the implications of the two strategies. Shallow porting is the easier route; deep porting offers opportunities to optimize application performance.

Stephe Walli of Softway – a company that markets a tool that assists shallow-porting UNIX applications – presented the argument in favor of shallow porting: it's easy. The shallow port of the Apache Web server, for instance, took an afternoon. Deep ports require an investment of both money and time. There must be some payoff in either functionality or performance to make this investment worthwhile. Walli pointed out that most companies don't port their products to NT to gain functionality; this leaves performance as the primary motivation to undertake a deep port. However, only the most resource-intensive applications have much to gain from code tweaking. Developers of most products would be better off taking the easy route.

For those who need to squeeze performance out of their applications, Ramu Sunkara of Oracle and Steve Hagan of Top End described their experiences deep porting database applications. Steve gave the encouraging news that 80% of the code will be a "nice port" and require very little programmer involvement. For the 20% of performance-critical code, the two offered the following advice:

- Avoid the C Runtime Library. The Win32 API is incredibly rich and will provide most of the tools you need to get your job done. Use the API directly; performing the equivalent CRT (C runtime) calls merely adds an additional layer of code.
- Use threads and fibers to maximum advantage. Windows NT is a multithreaded operating system, and the addition of fibers – lightweight, application-scheduled threads – gives the developer a high degree of control over CPU usage. Ideally, there will be exactly one active thread for every CPU on the system; this will provide maximum CPU usage without the overhead of context switches. Achieving this goal is

a tough challenge for the developer. Also, use the thread and fiber APIs directly instead of a thread package; as with the C Runtime Library, there is no reason to add an additional layer of code.

- Take advantage of the flexibility of the I/O subsystem. The NT I/O system gives immense power and flexibility to developers. For those fluent in the API, it is easy to bypass the file cache, perform asynchronous I/O, and hook your own code deep into the filesystem.
- Take advantage of the NT security model. This is an area that straddles performance and functionality; the NT security model is incredibly rich. By using it directly, you will not only lose a middle layer of software in your application, you will also gain additional flexibility.

BOFS

Blind Geese (Porting BoF)

Summary by Brian Dewey

Intel Corporation coordinated a birds-of-a-feather session about porting UNIX applications to Windows NT while maintaining good performance. Inspired by the name "birds-of-a-feather," they opened this BOF session by talking about geese. A flock of geese (they're a gaggle only when they're on the ground) flies 70% farther by cooperating in their V formation. The lead goose guides the flock and does a large amount of the work; a trailing goose, flying in the slipstream of another bird, experiences less wind resistance. When the lead goose tires, another takes over.

Likewise, Intel said, researchers would be more productive if they shared their experiences. This session provided a forum for that: an opportunity for members of the community to describe the problems they've had with NT performance and get help from fellow members. After a slow start, the audience warmed up and started raising many issues about NT performance. However, Intel's analogy, although cute, was too optimistic for such a young community. There was no "lead goose" for this BOF; nobody had the experience to give definitive answers for most of the questions raised.

Perhaps more problematically, NT is new territory for most of the audience; this impacted the quality of the questions. Only one person raised an issue and had numbers to quantify it: when doing a performance analysis of WinSock2, he noticed that using certain message sizes caused a dramatic reduction in throughput. He could find no pattern to predict what message sizes would be bad and wanted to know if anyone else had witnessed this behavior. (Nobody had.) Most of the other performance issues were based on vague impressions. Typical statements were "I noticed the file cache doesn't perform well when accessing large network files" and "My NT Server runs unexpectedly slowly when executing multiple interactive sessions." Without quantification, however, these issues are difficult to address.

In spite of all this, the BOF session had the right idea. People will encounter problems as they move to NT. The more opportunity to discuss these problems with other members of the research community, the smoother the process will be. I suspect that, in a year's time, members of the community will have gained the experience to ask good questions, give good answers, and make this BOF live up to its image.

Windows NT Futures

Summary by Brian Dewey

Frank Artale, Microsoft's director of NT program management, and Felipe Cabrera, architect in charge of NT storage systems at Microsoft, ended the workshop with a question-and-answer session about the future directions of NT. The session was informal. Artale's brief opening remarks seemed to be the only prepared part of the presentation. A single slide listing important topics in NT's future – storage, I/O, multiprocessors, clusters, memory, and management – dominated the projection screen for almost the entire talk. Once Artale put that slide up, he opened the floor to questions. And the questions poured in – so many, in fact, that an hour into the session, Artale and Cabrera were still fielding questions on the first topic.

So what information did the two panelists reveal about disk storage? First, they acknowledged that NT 4.0 has some deficiencies – for instance, if you ever try to run the `chkdsk` utility on a volume with thousands of files, you might find yourself waiting an uncomfortably long time (i.e., hours!). The NT File System (NTFS) fragments files more than most users would like. NT 5.0 will attempt to correct both of these problems: in the first case, by reducing the number of times a validity check is necessary and in the second by improving the disk allocation algorithm.

Second, Artale and Cabrera spilled a "technology piñata" upon the audience – a deluge of interesting but loosely related improvements to NTFS. Some of those improvements deal with loosening some of the constraints on NTFS present in NT 4.0. For instance, the new version will be able to support 2^{48} files, up to 2^{48} bytes in size, on a volume.

Additionally, NTFS will be able to support sparse files – say, if you had a megabyte of data distributed over a terabyte file, NTFS would store just the megabyte on disk. Further, NTFS in NT 5.0 will have several new features. The most frequently cited is content indexing; this will allow a user or a Web server to search a hard drive more efficiently. NT 5.0 will also sport symbolic links and a change journal that tracks modifications made to files (an essential component of the Coda-like filesystem).

A few thoughts struck me as I watched this question-and-answer session unfold. First, the audience seemed more interested in Windows NT than on any research being done on Windows NT. What else could explain the flood of questions about the unglamorous topic of disk storage? I don't think this phenomenon resulted from Windows NT merely being the common denominator of the diverse crowd.

To give my own biased evidence: I wasn't bored for one instant during this presentation, even though it came at the end of three days of diligent notetaking. While much of the credit for this goes to Artale and Cabrera's engaging style, I think the primary reason is that Windows NT is interesting in its own right.

Second, this presentation emphasized how there aren't many NT gurus outside of Microsoft. Sure, there are plenty of people trained in Windows NT, but the training seems to amount to knowing which menu option to check to achieve a desired result. Not many people know exactly what happens once that menu is clicked.

The UNIX world lives by a vastly different standard; not only is source code easily available, but there is also a large array of books that explain, in detail, the inner workings of the UNIX operating system. Without a steady supply of information, members of the research community are forced to wait for opportunities like this session to pose their questions to the insiders.

Large-Scale System Administration of Windows NT Workshop

SEATTLE, WA

August 14-16, 1997

KEYNOTE ADDRESS

From UNIX into Windows NT – A Long Day's Journey Into Night

David Korn, AT&T Laboratories

Summary by Gus Hartmann

David Korn is well known for his work on the Korn shell. Less well known is his work to make porting code to Windows NT from UNIX and vice versa easier for programmers. By his own admission, he has never administrated a computer system, so his dealings with Windows NT have been from a programming perspective. Ideally, he would like to see code being ported from Windows NT to UNIX and vice versa based on the needs of the program, and to that end, he has been working to increase the ease of migrating code. [*Editor's note:* Korn has his own article on page 28 in this issue.]

Before discussing Windows NT, Korn gave a brief overview of his computer experience in terms of platforms and programming languages. His list of platforms was divided into two sections; the first began with an IBM 650 and machine language to a VAX 780 running UNIX; the second ranged from Apollo Aegis and C to MVS Open Edition and ANSI C.

As a brief overview of Windows NT as a programming platform, Korn gave lists of the good and the bad qualities of Windows NT.

The good:

- file handles and file handle duplication
- thread model
- user interface
- search order for DLLs
- C/C++ Compiler
- I/O performance

The bad:

- complexity
- inconsistent programming interface
- lack of a shell
- documentation
- security model
- scheduling
- single-user mentality
- registry
- UNICODE
- frequent reboots

Several common myths about Windows NT were debunked by Korn during his address. Korn has investigated each, and found it to be without basis in fact. Among these were the following:

- Windows software works on all the Windows platforms.
- Everyone uses Windows.
- Windows is less expensive to purchase initially.
- UNIX is going to die.
- UNIX runs faster than Windows.
- Windows experts are plentiful.

TECHNICAL SESSION

Administration I – Enterprise Management

Summary by Steve Hillman

Three papers focusing on various aspects of administering a truly large-scale NT installation were presented in this session.

The first paper, "Domain Engineering for Mission Critical Environments," was presented by Chris Rouland of Lehman Brothers. Lehman Brothers is a large

Great Workshop Quotes (from Phil Scarr's Closing Remarks)

Have you ever heard of drag and UN-drop?

Iconic systems management is a fake!

Human efforts alone are insufficient...

Last time I checked, Office 97 was a productivity app, NOT an operating system!

Now booting the registry. . .please wait. ..

This really isn't as bad as it sounds. . .

NT 5.0 . . . I'm running it today. What I run, you don't want!

Has anyone noticed that [cable TV] Channel 29 is dedicated to the Windows 95 Blue Screen of Death?

Sysadmins always know what should be in the next revision. . .

Windows 95 gives you all the security you deserve. . .

My day job is installing and configuring Exchange at Boeing!

investment firm with offices worldwide and about 8,500 employees. It has roughly 3,200 SUNs, 7,000 PCs, and 400 NT and Novell servers. It runs both UNIX and NT servers, and still has some legacy Novell servers running.

According to Rouland, when setting up domains for NT servers, there are two models you can follow, and which one you choose is primarily dictated by the size of your network. Single-master domain models work in smaller environments. Lehman Brothers uses a multiple-master domain model, with a total of three masters – one in North America, one in Europe, and one in Asia. Multiple-masters have higher resource requirements, but function better in very large organizations.

Lehman Brothers found that it was good to throw money at problems. With lots of servers, it was able to separate functions onto different servers. This also made it easier to delegate authority for managing services. Sysadmins would be given access only to a server that ran a service they were responsible for.

Rouland stressed the need for highly available Domain Controllers. All of Lehman Brothers DCs run on Compaq servers with RAID arrays, ECC DRAM, fail over CPUs, etc.

One interesting point that Rouland raised: consider a service pack update to be an OS upgrade. Most SPs make changes to the kernel files and hence should be treated as OS upgrades, rather than simply as patches. They have found that their Domain Controllers are not happy running at different SP levels, so they must synchronize as much as possible SP and hotfix updates to their DCs.

Rouland also talked about namespace. There are four primary methods of managing names under NT:

1. WINS
2. LMHOSTS
3. DNS
4. Lanman Browsing

Lanman Browsing is broadcast-based and, in large networks, does not propagate well and gets confused. Microsoft DNS "works." The only realistic choice in a large NT installation is WINS; however. it still has its problems. One of these is that it has no decent administrative interface. Lehman Brothers found only one command-line interface into the WINS database, and it was a poor one. People there are working on their own WINS service based on the Samba code. [*Editor's note:* see Chris Hertel's article on page 23 about the University of Minnesota's extensions to Samba and WINS.]

The first question of the session was how Lehman Brothers handles account synchronization between UNIX and NT. Rouland responded that, currently, it doesn't. Accounts on each system are given the same username, but no password synchronization occurs. They're looking at a Kerberos solution in the future. As far as file space goes, people at Lehman Brothers have set up an NFS server on some of their NT servers and provided limited cross-platform support that way.

Lyle Meier of Texaco detailed a problem he was having with users in the US getting authenticated across a slow link from a PDC in Kuwait. He wanted to know if Lehman Brothers experiences similar problems. Rouland responded that they had. The short-term solution was to turn off net logon at the remote site. The long-term solution that they hope to implement is a rewritten WINS service that keeps track of hop counts for Domain Controllers and authenticates to the "closest" one, rather than the first one to answer.

Rouland had mentioned earlier that the Microsoft DNS server stored all of its information in the registry, preventing you from finding any `named.boot` or similar files. Eric Pearce of O'Reilly commented that, in his experience, the MS DNS could be set up to use files. According to documentation I've read, the MS DNS server can be set up to use files, but when doing so, the DNS Manager cannot be used; and if you switch to using the registry, you can't switch back.

Jeff Hoover of Cisco asked how Lehman Brothers intends to manage its browsing environment. Rouland said that people there are still trying to figure out how to deal with browsing issues in general – everything from turning it off completely, to using Samba, to turning it off on enough servers that they know which servers will act as browse masters.

The second paper, "In Search of a Complete and Scalable Systems Administration Suite," presented by Joop Verdoes of Shell International Exploration and Production, detailed his frustration in trying to find a suite that really managed their machines. This paper wasn't about NT management specifically, in fact, most of the examples given were UNIX-based. Unfortunately, SIEP has not met with much success so far.

Verdoes summarized many of the inadequacies he'd found in the management suites he'd dealt with and then listed features he believes a good suite should have. They include:

- storing, managing, and distributing config files – not just the basic config files, but *all* necessary config files
- workstation configuration management – a mechanism for obsoleting a workstation's old config files at boot time and automatically fetching the current ones, falling back to the obsoleted ones if the fetch fails
- decent logging
- version control – simple backout to a previous OS version if the current version has problems

On the bright side, Verdoes provided some good lines for the conference. Among them were, "Have you ever heard of drag-and-undrop?", referring to his frustration with not having an audit trail on most system management suites, and "Iconized management is a fake!", referring to windows full of several hundred icons, all of which look exactly the same.

The final paper of this session, "Large Scale NT Operations Management," was presented by Steven Hearn of Westpac Bank in Australia. Westpac has about 33,000 employees and 10,000 PCs spread out across 1,100 bank branches. Each branch runs its own NT 3.51 server, and each server speaks IP over 9600 bps modem links back to the headquarters. There's no NT expertise at each branch, so all server management must be done remotely over the 9600 bps links.

Because there's very limited bandwidth to each branch and limited staff at the operations center, Westpac has been focusing on trying to automate as much as possible. It's also been adding more applications at the branches, increasing complexity.

Hearn summarized some of the key challenges that his operations center is facing:

- technical (NT automation, capacity and performance reviews, and coordination of large data transfers to all remote sites overnight)
- logistics (coordinating data at remote branches)
- centralized security and account management (Westpac currently uses a separate domain for every branch.)
- managing the NT footprint (With every service and management tool added to each server, the resource requirements grow. With over 1,000 servers in place all over Australia, constant hardware upgrading is not an option.)
- change management and data currency (By "change management," Hearn is referring to some sort of audit trail that allows config changes to be tracked.)

Some of these challenges are already being worked on. Hearn's group is now using the User Manager, Print Manager, and Server Manager as remote management tools to manage their branch servers centrally. This provides a very basic level of management. They're presently working on the security of this – tightening up controls for who can remotely connect and what they can do.

For NT event logging, they're using SNA Server and passing events into NetMaster for a centralized view of NT alerts. They've had this system in place for nearly two years now, and it has helped them learn a lot about the volume of NT events that can be expected and what events should be passed along to their help desks for immediate action.

Westpac is currently using FTP for remote data distribution. Each night, the NT servers ftp into a central mainframe to get updates via FTP. They're currently able to disseminate up to 2Mb to all 1,000 branches each night.

The operations center is now reviewing several performance-monitoring software packages and working on in-house automation software that will alert the operations center when the automation fails. Their help desk training is also being improved to provide more problem resolution on the "front lines."

Questions started with Rob Wilkins of Dupont asking Hearn how he managed to remote boot his clients over their 9600 bps lines. Hearn explained that all booting happens within the branch. The PCs are diskless and connect via 10BaseT to the NT server at the branch. They get their disk images from the local server.

Till Poser of DESY asked why they chose to have a separate domain at each branch and how their centralized account management works with that many domains. According to Hearn, their remote-boot architecture required them to have separate domains at each location. They currently have no centralized accounts – accounts exist only at each branch, and when the servers are being managed from the operations center, the administrators must log in to each server.

Derek Simmel of CERT wanted to know what measures had been put in place to protect the confidentiality and integrity of data. Hearn answered that, for the most part, he couldn't say much under confidentiality agreements with the bank, but he did say that their remote FTP transfers go through a validation process to make sure the remote server is a valid server and the data have arrived intact.

TECHNICAL SESSION

Administration II – Integration

Summary by Steve Hillman

This session began with a paper entitled "Integrating Windows 95/NT into an existing UNIX Enterprise Network," presented by Cam Luerkens of Rockwell Collins Cedar Rapids Division.

In 1996, Rockwell Collins Cedar Rapids had roughly 25 UNIX-based application servers, 400 UNIX workstations, and 3,500 PCs running Windows 3.1 and PC-NFS. When it came time to transition to Windows 95 or NT on the desktop, management decided that the UNIX-based servers should stay on and that some form of PC-NFS client should continue to be used unless another solution could be proven to be better.

The transition team had several problems to consider:

- PC-NFS 5.0 was a DOS-based NFS client. A new solution for file sharing would need to be found.
- Authentication was strictly NIS based before. Now NT domains were potentially being added
- Printing

To solve the first problem, eight NFS and two SMB products were evaluated. Based on evaluation in several key areas, Hummingbird's Maestro NFS client narrowly proved to be the best product and was adopted sitewide.

Authentication was handled by having the PCs authenticate to an NT domain. Initially, the NT and UNIX passwords were just set to the same thing and then given to the users. Hummingbird then agreed to write password synchronization routines to sync 95 and NT passwords to NIS. The NT domain routines were added last April, and the 95 routines were finished just before this conference and are currently being tested.

To handle printing, the NT workstations were set up to print to NT servers. Because NT connection licenses weren't purchased for them, the Windows 95 machines were set up to print to UNIX via Maestro.

A Perl login script is run at the Win95/NT workstation whenever a user logs in. This handles setting up the user's environment, ensuring that, regardless of where the user logs in, the environment will be the same. Additionally, a Rockwell programs group was added to the Start menu. Each icon in the group actually points to a Perl script that sets up the environment before running the application – if the application has never been run on the workstation before, appropriate installation steps are taken automatically.

When implementing the rollout, the transition team found that the first 300 users went without any major problems. When they hit 300–500 users, they started having problems with file locking on the NFS servers when running the Perl login scripts. This turned out to be a Maestro issue and was eventually resolved. At 700–800 users, they had problems with load: all users were running their Perl scripts from the same server. Using a combination of NIS tables and more Perl, they were able to load balance the clients across several servers based on what subnet they were connecting from.

The question period began with Rob Elkins of Dupont asking why Samba was not used. Luerkens answered that management required that any product they use have available support they can buy. The transition team would also have had to show that Samba was significantly better than an NFS solution, because NFS was already in place and working well.

Steven Hearn of Westpac Bank asked about virus scanning. Luerkens said that the PCs use realtime virus scanning to block most viruses, including macro-viruses. Rockwell is also running a version of McAfee's antivirus software on its UNIX servers. This gets run each night and automatically generates email notification to any users found to have a virus within a file in their file space.

Ian Alderman from Cornell University asked for some more information on Maestro. According to Luerkens, Maestro consists of two parts – the client, which runs on NT and 95, and the NFS daemon, which runs on the UNIX servers. The daemon ties into NIS for authentication. The Win95 password-changing program is currently being tested. It will allow a password changed on a Win95 client (using the standard Win95 password-changing interface) to be changed in both the NT Domain Controller and the UNIX NIS maps.

Jim Dennis wondered why file locking was an issue if the Perl scripts were mounted on a read-only volume. As was mentioned earlier, this was a bug with Maestro – the NFS daemon was locking files on reads as well as writes.

The second half of the technical session consisted of a panel discussion entitled "Management Integration/Politics." The panel moderator was Boris Netis. Panel members were Paul Evans and Phil Scarr of Synopsys and Chris Rouland of Lehman Brothers. This panel focused on the politics of bringing Windows NT into an organization.

In the case of Synopsys, a vocal group of senior VPs has been pushing for several years to have the largely Mac-based company switched to PCs. When Win95 was released, the sysadmins were able to head off this push, but when Apple's stock started to slip in early 1996, and with the pending release of NT 4.0, the company started planning the switch (much to the protest of the Mac support staff!). In the early stages of planning, the CEO asked why they couldn't just come in one weekend and replace all of the Macs with PCs running NT 4.0. The MIS director nearly leapt across the table and throttled him.

At Lehman Brothers, NT crept in through the back door. It started out as a pilot project. The first business unit to get NT on the desktop was the brokerages. This was in part due to the relatively simple requirements of the brokerage users. From there, they started planning how they would introduce it into the other business units. At the same time, Microsoft came up with a true 32-bit version of Office. One of the driving forces behind the NT rollout was the richness of the applications available for it. With the release of a 32-bit Office, the drive to get NT onto every desktop increased. The behind-the-scenes database servers that keep everything running are still largely UNIX based.

Evans touched upon the politics involved in replacing users' desktop environments. Users may tend to hold you responsible even if it wasn't your decision to replace them. Some thought needs to be given to how the changes will affect the user.

Joop Verdoes asked why it was necessary to pick a single platform for the entire company instead of giving the users what they want. The consensus from the panel was that most companies can't afford to support this. At Synopsys, it was a requirement that the support staff know exactly what was on everyone's desk to ensure that support was actually feasible. Rouland explained that at Lehman Brothers, each business unit is responsible for choosing its desktop. They don't decide technical details, but they decide whether to stick with their legacy Win3.1/Netware system or upgrade to NT. If they choose NT, Lehman's support staff handles all other aspects, asking the end-user/managers only whether they need a "fast computer, really fast computer, or really, really fast computer."

John Holmwood of Nova Gas commented that he's being asked by senior management to justify why they need to worry about integrating NT and UNIX. He asked the panel if they'd faced the same questions. Netis commented that there are good reasons for integrating the two platforms. One is avoiding duplication of services: there's no need to waste effort setting services up under NT if they already work perfectly well under UNIX. A second reason is sharing file space. If an organization needs to move files between the two platforms or has a large investment in UNIX disk space, it makes sense to try to integrate the two.

Scarr commented that it would be good if integration can happen at a political level as well as a technical level. Synopsys, has a completely separate NT administration group that doesn't even work in the same state. The NT group has no idea what the UNIX group is doing . When the UNIX group installed NISGina (a freeware product that allows an NT machine to authenticate to a NIS server at login time), the NT group wasn't interested.

A comment was made by Crystal Harris of the University of California at San Diego that not all environments are so homogenous. In a university environment, in most cases, the users dictate what platforms get used, often resulting in virtually *every* platform being used! She is very interested in seeing programs that will help integrate such a heterogeneous environment.

Steve Hearn asked the panel to comment on the integration issue of a single logon. Rouland responded that Lehman Brothers has been looking, so far to no avail, for a commercial solution to this problem. Its group is considering attacking it from the other side – dropping in LDAP-NIS gateways and making the UNIX boxes authenticate from the NT Domain controllers, rather than the other way around. Scarr expects that, within 18 months, Synopsys will be almost entirely NT based, and this will be a nonissue.

Dave Ellis of Sandia National Labs asked if senior management was aware of the benefits they were getting with NT, such as increased security. Evans responded that, in his opinion, no. The move to NT was largely market driven – "Everyone else is, so we have to too." The Gartner Group's Total Cost of Ownership reports are also motivating management to move to NT. The subtleties of the various operating systems are lost on them.

Joop Verdoes wanted to find some decent capacity-planning tools that would allow him to determine how many servers he'd need to deploy. Netis said that Microsoft had some simple tools that would help. Rouland said that he had seen no decent tools. He just grossly overengineers his network. Scarr asked for a clarification of what a "server" is. At Synopsys, most of the servers are legacy servers and are UNIX based.

A participant then brought up the issue of cloning NT boxes to facilitate quick rollouts. He currently spends about eight hours on each NT machine he sets up because he can't just clone the hard drive because of the Security ID issue. Apparently, the panel was not aware of this issue because they've all been cloning machines at their sites. As was explained by several people in the audience, if you duplicate the SID, the effects can be quite subtle at first, but duplication can cause problems with SMS and with browsing on large networks.

It will also wreak havoc if the machines are upgraded to NT 5.0, because NT 5 uses the SID as an identifier in its directory services tree. New SIDs are not created when a machine is upgraded.The unique SID is not generated until the machine is rebooted after the initial text-based installation. If you switch the machine off, rather than reboot, and then clone that hard drive, you'll have SID-less PCs that are ready to be configured. This still leaves a lot of steps in the installation process that must be completed on every clone.

Kimbol Soques of AMD has spent a fair amount of time working on this problem and has a lot of information accumulated from various sources. She offered to email the info to anyone interested. Her address is <kimbol.soques@amd.com>.

John Crane asked the room how many shops had separate UNIX and NT system administrators. The vast majority had separate administrators.

Vince Kluth of GDE Systems compared the information model to a type of government, where a university could be compared to a democracy – each user in control of their own machine. A corporation is more like a dictatorship, where a single voice decides what machines shall be used everywhere.

The last question of the session concerned experience with Mac support on NT servers. Netis commented that he'd had some experience with it and found it to be mediocre. Scarr relayed the all-too-common story where a user sets up an NT server, tries to set up a queue for an Appletalk printer, selects "yes" to the question "Do you want to capture this printer," and then wonders why the printer disappears.

Rouland added that one of Lehman Brothers departments – Creative Services – has Macs, and when an NT server with Mac File and Print services was set up for them, they loved it.

INVITED TALK

MS/Zero Administration Windows

Dan Plastina, Microsoft, Inc.

Summary by Gus Hartmann

In the next version of Windows NT, Microsoft is attempting to reduce the total cost of ownership by lowering the administrative costs. This would ideally involve remote management of software, users, and hardware profiles. Dan Plastina of Microsoft presented an overview of the new administration features that will be coming in Windows 98 and Windows NT 5.0.

One of the key new features is Intellimirror functionality. Essentially, this consists of mirroring every write to the local disk to a network disk. This will allow complete re-creation of the local disk in the event of a catastrophic hardware failure. This is dependent upon the presence of a boot ROM containing a DHCP client that is a part of the new motherboard standards on which Microsoft is working with other industry players to develop. Upon booting, the client machine connects to a Remote Boot server and updates the local hard drive as necessary to make it identical to the user's home drive. This updating could involve installing certain user applications or even installing a different version of the operating system.

Seamless roaming will also be enabled, allowing users to sit at any machine and, as far as hardware permits, have a re-creation of their normal environment on the remote machine. This will include any applications that would be installed locally, as well as user documents.

As a means of installing applications, remote installation based on document type will also be supported. For example, a user receives via email a document in a format that the user cannot open with the software that is installed locally. The client machine would indicate to the server that an application to deal with that file type is needed, and the server would remote install a suitable application, providing support to the user without human intervention. [*Editor's note*: The server would then make an electronic funds transfer to pay for the new application license to Microsoft. Just kidding!]

Another important part of this initiative is the Windows Management Services, such as Web-Based Enterprise Management (WBEM) and Windows Management Interface (WMI). WBEM allows sharing of management data between network devices, desktops, and systems.

Also featured will be new modular administrative tools, called "snap-ins." By selecting only the tools necessary for their site, administrators can build a tool set suited to their needs. Application deployment will be improved for administrators, enabling a system administrator to push applications onto a client machine by "assigning" the application to a given user. Other software packages can be made available without forcing installation by making them "published" for a given user or group of users. "Published" applications can be added via the "Add/Remove Programs" commands in the control panels.

TECHNICAL SESSION

Tools

Summary by Steve Hillman

This session contained three papers. The first paper, "U/WIN – UNIX for Windows," was written and presented by David Korn. U/WIN is a set of libraries and header files that allow UNIX source code to be compiled and run on a Windows NT or 95 system with little or no modification. The software was developed by the Software Engineering Research Department at AT&T Labs, with most of the work being done by Korn. The binaries are freely available for noncommercial use.

In writing the software, the team's goals were

- minimal source code changes required
- minimal performance overheard (compared to the native NT function calls)
- ability to mix and match UNIX and NT functions
- NT and 95 support
- availability

These goals were met, either partially or completely. Some of the problems they had to overcome included

- case-insensitive filenames and filename limitations in general
- pathname syntax
- <nl> vs. <cr><nl>
- signal handling
- incomplete interface
- permission mapping

Korn did not go into all of the technical details on how some of these problems were overcome (he only had 30 minutes!), but a lot more detail can be found in his paper.

U/WIN was written using the Win32 API. Because this API is present on both Windows NT and Windows 95, most code compiled to use U/WIN will run on either platform (note that some Win32 API calls are not available on Win95 and hence can't be made available via U/WIN on Win95). Virtually all UNIX calls were implemented in U/WIN. Additional features include:

- `stdio` calls based on `sfio`
- Microsoft C library used for remainder of calls
- support for hard/symbolic links
- memory mapping and dynamic linking
- System V shared memory
- BSD sockets implemented on top of Winsock
- `setuid/setgid` support on Windows

The current version of U/WIN, 1.30, was released just before the conference. It includes a C compiler wrapper for Visual C/C++. To date, over 170 tools have been ported to NT using U/WIN, including `ksh-93` and `vi`. Development team testing has produced the following results:

- no loss in I/O performance using U/WIN
- `fork/exec` three times slower (than on Linux running on the same machine)
- `vfork/exec` ~ 30% slower
- `uwin_spawn` 25% slower
- deletes two times slower (than deletes done in native NT)

The U/WIN project is still being actively worked on. In the future, Korn hopes to add these features:

- case-sensitive file names
- a simulated mount table
- the rest of the inet daemons
- 64-bit file support
- POSIX pthreads API
- a registry filesystem (making it possible to `cd` to a folder in the registry)
- `tksh` and `deet` debugger (Currently, debugging can be done only from the NT console.)
- I18N based on UFT8 (Unicode)
- n-DFS
- multiformat documentation
- run SCO/Linux binaries

Korn provided some URLs that are relevant to this paper:

Graphs from this session and the keynote
<http://www.research.att.com/~dgk/usenix>

U/WIN binaries
<http://www.research.att.com/~dgk/sw/tools/uwin>

David Korn's sysadmin survey
<http://www.research.att.com/~dgk/survey>

Home of the Korn shell
<http://www.kornshell.com>

Phil Scarr asked Korn if he'd compiled `Gcc` under U/WIN yet. He said he hadn't, but that it should be a trivial task. In his opinion, most programs should compile with virtually no modifications, and he's hoping "we" will get to work on that!

Jim Dennis asked if he'd ported `rdist` yet. Korn explained that he started by porting just the minimum required to make a usable system. Not being an administrator, he hasn't ported `rdist`. He's eager to have someone else do it though.

An audience member asked Korn to elaborate on his comment in his paper that "there appear to be few if any technical reasons to move from UNIX to Windows NT." Korn responded that this may be more of a personal preference than anything else – he likes the Linux environment for developing software and doesn't need the GUI that NT offers. Without the GUI, he said, there's little other reason to move to NT.

The second talk, "Utilizing Low-Level NTFS Disk Structures for Rapid Disk Searching" was by Michael Fredrick of Pencom. There was no associated paper with this talk.

Fredrick started with a description of the problem: come up with a method for doing fast file searches that can search on various criteria such as date, file size, or file owner (for quotas). His primary interest in coming up with these routines is to develop a quota system for NT. There are products on the market that do quotas on NT, but apparently they work strictly on a file's position in the tree, not on the file's owner, making it impossible to determine how much space a particular user is using.

According to Fredrick, filesystems normally have special "header" files that store file attributes such as name, owner, size, and security. A search algorithm can use these headers for fast searching on attributes.

NTFS uses the MFT (Master File Table) to store this info. Unfortunately, the MFT is a database file, not a sequential file. It's also completely undocumented and cannot be opened with the Win32 API. This makes it rather difficult to use it for fast searches. To view the MFT at all, Fredrick had to write routines to open an NTFS volume in raw mode and seek to the location of the MFT on the disk (it always starts at the same place), then read it in raw.

Fredrick explained that rebuilding a map of the volume from the MFT is not a trivial task. It uses a very elaborate (and highly undocumented) structure for storing the layout of the directories. Additionally, not all information is resident in the MFT – it can contain pointers to other locations on the disk. In fact, NT treats everything to do with a file as an attribute – the name, date, security ACL, even the data itself. If an attribute is too large to fit in the MFT, the MFT just contains a pointer to it elsewhere on the disk. A side effect of this is that very small files (typically under 1k) can fit entirely within the MFT and do not require any additional disk access at all.

Unfortunately, this is more of a work in progress than a finished product. Fredrick does not yet have any usable routines to do this fast searching.

Fredrick listed some references for anyone interested in more information about the NT filesystem:

- *Inside Windows NT* by Helen Custer. By all reports, this is an excellent reference and the only one released by Microsoft. The only thing this book lacks is the numbers (such as offsets, sector locations, etc.).
- Martin Von Lowis's Web site: <http://tiger.informatik.hu-berlin.de/~loewis>. Martin helped develop for Linux the NTFS driver that contains a fair amount of documentation within it. You can get the source from here, as well as some pointers to other useful sites.

In response to Fredrick's comment about having to rebuild pathnames by walking backwards through the MFT, Rik Farrow commented that it might be worth considering the algorithm that UNIX's `ncheck` uses where it builds a directory tree as it scans through the i-nodes. The same approach could be used while doing a search through the MFT.

The last paper, entitled "Adding Response Time Measurements of CIFS File Server Performance to NetBench," was presented by Karl Swartz of Network Appliance.

NetBench is a software package for testing throughput of CIFS (aka SMB) fileservers. It doesn't include response time measurements though, so Network Appliance set about adding them. This was done using hardware – a packet capture device was added to capture a trace of a client talking to the server. Then the trace was analyzed to figure out how long it took the server to respond under various loads.

Not surprisingly, when Swartz compared a Network Appliance F630 to a Compaq ProLiant 5000 with hardware RAID, the F630 performed better. A detailed comparison can be found in the paper.

Mark Maxwell asked if there was any write caching being done on the Compaq server. Swartz responded that the intelligent RAID controller they were using does some write caching. Also, because the Compaq was using NT as its OS, NT does a substantial amount of write caching itself.

Phil Scarr asked if they were really doing a fair test when they were comparing a 500MHz Alpha box to a quad-Pentium Pro-200 box. Swartz didn't think the servers were processor bound, though, and because the rest of the hardware was very similar, should have made for a fair comparison. Scarr still had some concerns and suggested that a much better comparison would have been between a slower Network Appliance and a faster Compaq. This would truly have shown that CPU speed was irrelevant.

WORKS-IN-PROGRESS SESSION

Summary by Steve Hillman

The Works-In-Progress (WIP) session is made up of a number of very short presentations detailing projects that aren't quite ready to be presented in the form of a paper yet. A total of eight WIPs was presented.

The first WIP, "DHCP in Eight Days," was presented by Phil Scarr of Synopsys. Scarr chose to use ISC's `dhcpd` running on UNIX. DHCP leases are keyed by the Ethernet addresses of the clients. ISC's DHCP server stores its lease information, including the name of the client (which gets sent to the server during the DHCP handshake) in a text file. With this text file, the DNS can automatically be stuffed with the IP address and hostname of machines that have leases. Scripts that run every few minutes were set up to check for changes in the leases text file and push them into the DNS.

One drawback to this approach is that every PC must be uniquely named to prevent DNS conflicts. Because the engineers insisted on having administrator access to their personal machines, it was impossible to guarantee that machine names didn't change.

An audience member asked how Scarr dealt with the Network Operators staff who generally want every machine to have a static IP address. Scarr answered that he just told them, "Tough. The NT project needs DHCP." With the automatic updating of the DNS, though, it becomes more or less a nonissue – the PCs are named using a specific naming convention, and with the name in the DNS, it is easy to identify the location of problem machines.

For more information on ISC's `dhcpd`, check their Web site at <http://www.isc.org/>.

The second WIP was on WINS and was presented by Chris Rouland of Lehman Brothers. Rouland's group has found that the Microsoft WINS server has a fairly high failure rate. The MS WINS server is based on a JET database, and there's no programmatic interface to it. Because Microsoft provides no good tools for managing WINS, if the WINS server crashes and corrupts the database, you either have to restore from tape or wipe the database and start from scratch. For these reasons and others, people at Lehman Brothers are working on writing their own WINS server that's much more like DNS – a single primary server that pushes changes out to secondary servers, with all servers being read-only to clients. Their implementation is based on `nmbd`, a program that's included with the Samba distribution.

Chris Rouland tagged off to Chris Hertel of the University of Minnesota, [*Editor's note:* An article about Hertel's project appears on page 23 in this issue.], who also did a WIP on WINS. The University of Minnesota has a DNS with 40,000 entries and is currently running Samba. When Chris and his team tried to load the 40,000 DNS entries into Samba's `LMHosts` file, it took over an hour to launch Samba. They decided to build a back-end database for Samba's `nmbd` instead. They're also working on external references – telling `nmbd` to, for example, check with the DNS before allowing a WINS entry into the database. Finally, because the University of Minnesota, like most universities, is directly on the Internet, they're working on adding filters to `nmbd` to control who can and can't make queries and generate WINS entries.

Someone asked how each speaker dealt with `dhcp` putting "dhcp-" in front of all hostnames. Hertel said he wasn't familiar with this problem. Rouland said he hadn't experienced that and suspected it may be a configuration issue.

Todd Williams of MacNeal Schwendler asked all three speakers how they dealt with "renegade" DHCP servers on their networks. At the University of Minnesota, this has not been a big problem; routers are configured to automatically forward DHCP requests to a specific machine, so renegade machines can affect only their local segment. A good relationship with departmental LAN administrators helps here. At Lehman Brothers they use network stormtroopers. At Synopsys, they can black-hole the renegade's packets at the routers, but this doesn't protect the segment that the renegade is on.

The next WIP was on cloning NT workstations and was presented by Kimbol Soques of AMD. Her group needs to roll out over 6,000 workstations during the next three years. To do this, they developed a system based on cloning the hard drive of a completely configured machine. Then they discovered that this is not supported by Microsoft because it results in identical security IDs (SIDs) on every machine. This will mess up SMS on large networks and will also prevent NT 5.0 from working properly. This is not well documented.

They then discovered that the SID is assigned during the installation process when you switch from character-based to graphical-based install. Essentially, NT copies all necessary files to the disk, then wants you to reboot and proceeds to come up into the GUI where the configuration happens. If, instead of rebooting, you shut down the computer and remove the hard drive, you have a SID-less half-installed NT disk that you can clone.

Rob Elkins asked if one could use Ghost to clone the SID-less disks. Kimbol responded that Ghost or any other duplicating tool could be used.

Ken Rudman asked how applications were installed in this process. Kimbol said that several tools, including `sysdiff`, unattended `install`, and SMS installer, were used to install applications. No one way works for every application. Some applications, such as MS Office (surprise!), refuse to be installed in this manner. They had to resort to a network install for Office.

Ian Alderman commented that Microsoft Knowledge Base article Q162001 contains a summary of this information. Kimbol mentioned that the Resource Kit tool `getsid` prints out the SID of the local machine and allows you to determine whether you have duplicate SIDs. She also said that she had 20 or 30 other KB articles that referenced this problem. She'll email all the info to anyone who wants it. Her email address is <kimbol.soques@amd.com>.

Chris Kulpa asked if they were using KiXtart to do registry modifications as part of their remote installations. They are. For more info on KiXtart, check out <http://netnet.net/~swilson/kix.html>.

The next WIP, on Samba, was presented by Jeremy Allison of Whistle Communications. Allison is now the chief architect of Samba and filled us in on the features of release 1.9.17 (which should now be available) and on the features to come soon. [*Editor's note:* See Allison's article on page 16 of this issue]

Features of release 1.9.17 include:

- greater stability
- improved speed with shared memory
- browsing
- internationalization
- better documentation
- roaming profiles

Features that Jeremy hopes to have added soon include:

- GUI `config` tools
- opportunistic locking
- blocking locks
- dynamic loading of client code pages
- support for large-scale name databases (see the previous WIPs by Rouland and the article by Hertel)
- NT security integrated

Features that will take longer to add include:

- full Unicode support
- DFS support
- packet signing
- WINS replication between Samba servers

- Domain Controller protocols (even further out)

Till Poser of DESY presented the next WIP, entitled "Designing Application Support Infrastructure for a Heterogeneous Scientific Environment." He wins the "longest WIP title" award. DESY is the German High-Energy Physics Lab. There is a very mixed environment there, with predominantly UNIX and Win3.1 workstations and NT workstations just emerging. DESY is made up of three user groups: engineers, physicists, and administrators. The engineers are, for the most part, the most demanding and are becoming the early adopters of NT. Till, working in the Computing Services Department, is finding that with the proliferation of powerful desktop machines, CS is struggling to hang on to its central control of services and is having to rethink its service structure.

To help control things, they've adopted a three-tier, color-coded computer scheme that was originally developed by Remy Evard at Argonne Labs. A "green" PC is fully administered by CS and is suitable for most administrative staff. A "yellow" PC is built, installed, and configured by CS, but is left to the individual to administer. A "red" PC is under the sole control of the end-user. CS neither administers nor supports it.

CS is using a product called NetInstall to handle automatic installation of applications on demand. User accounts are centrally administered, and users have roaming profiles that allow them to log on from different workstations. NetInstall deals well with machines that are not all identically configured, which makes it a nice fit in their rather anarchic environment.

CS is also experimenting with using SMS installers to push out critical updates such as OS updates and service packs. Most of these services are strictly for the "green" machines to allow CS to maintain central control. CS is not sure yet whether they'll be able to apply these techniques to the "yellow" machines.

An audience member asked what the current distribution was among the three colors and what they expected the distribution to be in the future. Poser answered that, currently, all machines are "yellow" – most administrative staff are still on Win3.1. Poser expects about 25–33% will eventually be "green" machines. Very few will be "red" because "red" machines are not allowed to participate in the NT network.

The next WIP, "Integrating NT Workstations with Solaris," was presented by Gerald Carter of Auburn University. When NT was released, Carter's group's goal was to integrate it into the university's environment without having to install any NT servers. This meant that there had to be some way to handle NT's authentication requirements. They either had to duplicate their existing UNIX accounts on the NT workstations or somehow have the NT boxes authenticate via UNIX. They chose the latter. Luckily, there was already a freeware product called NISGina that did exactly that. NISGina is a DLL that replaces MSGINA.DLL and allows an NT workstation to function as a NIS client. NISGina adds the ability to pull home directory info and registry settings from the NIS maps during login. The login procedure goes like this:

1. The user presses `ctrl-alt-del` to log in.
2. The user enters username and password.
3. NIS maps are searched for username. If it's found, passwords are compared.
4. If the password matches, a LOCAL account is created on the system with registry settings pulled from the NIS maps. The user is then logged in under that account.
5. If the username is not found in the NIS maps, NISGina checks the local administrators group to see if this username belongs to it. If it does, the user is let in (assuming the password is OK).
6. If the username is not in the NIS maps or local admin group, the user is denied access.

Password synchronization is one-way. If you use the NT password-changing mechanism, NISGina changes the password on the account that was created locally and then uses `yppasswd` to change the NIS password. Unfortunately, if users change their passwords on a UNIX box, the local password does not get updated. Logins still work, because NISGina always looks to NIS for passwords at logon, but apparently this causes problems with shares because NT looks locally for those passwords.

Phil Scarr commented that even if you don't use NISGina for authentication, you can use its tools to query NIS maps.

The final WIP, "Measuring NT Performance," was presented by Steve Hearn of Westpac Bank. Essentially, Hearn was charged with accurately measuring NT performance to allow for capacity planning. With over 1,000 servers deployed, this was not an easy task. Hearn's group had to determine how many servers to baseline, at what frequency to poll them, and how many of the hundreds of performance parameters were actually useful. When selecting performance-measuring products to evaluate, they wanted to make sure that each product

- had little or no impact on the network
- had a 32-bit architecture
- could easily export the data
- must be easy to customize which metrics to measure
- must be easy to use

In the end, they chose two products to evaluate: Performance Works for NT by Landmark Systems and BEST/1 by BGS

Systems. They chose a small, basic subset of NT's performance parameters to baseline and decided that they would baseline them again after three months to look for significant changes.

Hearn then asked the audience if anyone else had done anything like this. A spokesman for HP said that they had internally used a product that they marketed called Measureware. This is apparently a snap-in for HP Openview that runs agents on each monitored server (both UNIX and NT). The agent gathers and analyzes data locally and then ships the results back to the Openview module.

Alan Epps asked which product was evaluating better so far. Hearn said that Landmark's Performance Works was working out really well. However, he noted that both companies were relatively new to the NT field and that these two products were still the best two he'd found so far.

After 90 minutes of very speedy talking, the WIP session came to an end. It will be interesting to return next year and hear these WIPs as full papers after the authors have worked out all the kinks.

INVITED TALK

MS/Security in Windows NT 5.0

Peter Brundrett,
Microsoft Corporation

Summary by Gus Hartmann

Peter Brundrett, a program manager at Microsoft, spoke about the future of security. Windows NT 5.0 should feature highly improved security, including data privacy on the desktop and on the wire, single enterprise logons, and decentralized administration for large domains. To implement these features, Microsoft will be utilizing strong network authentication, public-key credentials, and standard protocols for interoperability. Security administration will be greatly simplified, with unified account information for each account in a domain, fine-grain access controls, and per-property auditing. Most importantly, the new directory services will allow greater ease of administration through the hierarchical layering of domains.

Most notably, Kerberos 5 will be replacing NTLM as the authentication method for NT-based domains. However, the interoperability between existing Kerberos 5 distributions and NT has not been clarified. Kerberos 4 will not be supported. [*Editor's note:* See Ts'o's article on page 8.]

Another important new feature closely associated with the hierarchical layering of domains will be the "nesting" of groups within other groups. Circular groups will be allowed, and error-checking will prevent endless looping when checking for group membership. Local groups will also be eliminated, being made redundant by the new group format.

PANEL

Windows NT Security

Summary by Rik Farrow

The security panel, moderated, by Mike Masterson of Taos Mountain Software, consisted of Bridget Allison of Network Appliances, Jeremy Allison of Whistle Communications (Samba), Peter Kochs of Digitivity, and Peter Brundrett, Microsoft program manager.

The panel started with a quick survey of the audience. Ten percent of the audience used NT 50% of the time or more, while 80% used UNIX 50% or more. This served to identify the audience's background. Most of the audience claimed some experience with NT, and only 2 (out of approximately 320 people) admitted to using Microsoft's DNS.

Bridget Allison began by passing out "Ten Things Every NT Administrator Should Know About Security," summarized here:

1. By default, NT assigns Full Control to Everyone for newly created shares (Full Control is similar to owning a file in UNIX).
2. Password hashes in NT have no equivalent of UNIX's `salt`; the hash algorithms (DEC and RC4) are faster than the UNIX algorithm, making password cracking easier.
3. No file is safe from Backup/Restore (user) rights.
4. The FTP server distributed with the TCP/IP tools exports the entire disk partition (not restricted, as in UNIX anonymous FTP).
5. Registry settings, as delivered with NT, may have weak or inappropriate (for true security) access control lists (ACLs). Also, prevent remote registry modifications by setting the the following binary key to 1: `HKEY_LOCAL_MACHINES\System\CurrentControlSet\Control\SecurePipeServers\WinReg`.
6. Disable the Guest account (automatically enabled on NT 3.x anytime someone fails to login three times!).
7. Default permissions on `%systemroot%\system32` and `%systemroot%\system` are inadequate (similar to having `/` and `/etc` `777` mode under UNIX). Of course, fixing this will break some applications, such as MS Office.
8. Configure NT RAS (Remote Access Service) to use CHAP with DES (for authentication) and RC4 for link encryption.
9. Use NT auditing and monitor these log files (Be careful to select which events to audit.) Frank Heyne's event logging tools are available at <http://rcswww.urz.tu-dresden.de/~fh/nt/eventlog/index.html>.
10. For Internet accessible NT servers, unbind Server, Workstation, and NetBios (involved in SMB file shar-

ing) from the TCP/IP protocol, and block ports 135 (MS RPC, very dangerous), 137 (WINS), 138 (SMB over UDP), and 139 (SMB over TCP).

I admit to ad-libbing by adding comparisons to UNIX.

Peter Kochs talked a little about his company's (Digitivity's) product: a server-based sandbox that runs Java and ActiveX applets while displaying the output on or fetching user events from desktop browsers. The applets run within a controlled environment on the server, isolating the desktop's browser from security lapses and providing central control of applet execution.

Jeremy Allison mentioned that man-in-the-middle attacks against the SMB protocol are fixed with SP3 (service pack 3) for NT 4.0, but this breaks backward compatibility. Essentially, by capturing and retransmitting an SMB authentication (prior to SP3), you can easily discover the password used by a remote client. The patch (SP3) means that older systems (Windows 3.1 and Windows 95) will not work with patched servers.

Allison also said that the system call interface to NT is completely undocumented. This point had been made by other speakers, including some Microsoft employees. The reason given by Microsoft representatives is that the NT kernel changes faster than documentation writers can keep up with. Essentially, the source, some 16–20 million lines of code, is the documentation (the contents of 18 CD-ROMs).

Peter Brundrett of Microsoft mentioned that the conference is part of the process of improving NT. He suggested staying on top of service packs for security. He also said that there are some 2,600 system service APIs (UNIX has 157 system calls in 4.4 BSD). Security was not designed to be a plug-and-play feature, and there are security APIs that have not been documented, said Brundrett.

After this introduction, questions from the audience were accepted, starting with Chris Hertel of the University of Minnesota. Hertel stated that he had heard that the level of cooperation with MIT on Microsoft's implementation of Kerberos 5 was "not so rosy" and that his university planned to block access to Microsoft services at the router. Brundrett responded by saying, "If you are interested in open systems, that is a decision for you to make.... Our implementation will interoperate with Kerberos 5."

Hertel that said that he would like to point an NT workstation at a UNIX Kerberos server. Brundrett said this was working in Microsoft's labs. I contacted Ted T'so, who works on the Kerberos project at MIT. His response appears on page 8 in this issue, but the disagreement centers on Microsoft extending the defining RFC (1510) as a result of some vagueness in the text. In particular, how Microsoft will distribute information about groups appears to be in dispute.

Dave Ellis of Sandia Labs asked about DCOM (a distributed object system) configuration and got an answer that Microsoft was working on the problem.

Someone complained about the lack of interoperability between Windows 95 and NT (NT RAS cannot change password, and Windows 95 users cannot include a password with the "net use" command). Todd Needham, a Microsoft evangelist, said that this was a valid point. However, you need to choose the product that meets your requirements. "Windows 95 gives you all the security you deserve," said Needham.

Derek Simmel of CERT/CC said that they planned to start producing advisories for NT but that CERT does not have the level of expertise to support NT. CERT has been working on this for six months. His question was about validity and verification testing of the NT kernel. Brundrett responded by saying that Microsoft has tools to do this. Simmel asked if there are probe tools such as SATAN for NT. Brundrett said that someone else should do this (are you listening, Farmer and Venema?).

Jim Duncan, a systems manager at Penn State, said that Sun, IBM, SGI, and most other vendors worked with FIRST (Forum of Incident Response Teams), but Microsoft has been a glaring absence. Brundrett said that he would see about working with CERT and FIRST.

Someone asked if a user's role can be modified in NT 5. Brundrett said there will be an su command in NT 5.

Another person asked if it would be possible in NT 5 to prevent certain accounts from being added to particular groups (a way of preventing some object access exposures). Brundrett said it is an interesting idea. (Users administer to groups they create, which can lead to vulnerabilities in NT.)

The same person asked about control over encryption in network links. Brundrett replied that Microsoft is considering the notion of zones that control the security policy for encryption. Finally, the person asked if the Management Console was an extra cost option. The answer is no, but snap-ins (add-ons) are customizable extras, and you can buy them.

Like the panel that discussed source licensing of NT, the security panel produced a lot of heat, with most of it directed at Microsoft. Security and vendor secrecy do not mix well. But the Microsoft representatives did make an effort to answer questions and at least said that they would look into making changes, such as better participation in generating security advisories or working with FIRST. Altogether, it was an interesting and useful panel.

T.J. Fiske of Qualcomm said that his company had hundreds of "blasted" (cloned or ghosted) NT installations with the same machine id and asked what happens when we upgrade. Brundrett said that the SIDs do not change during an upgrade to NT 5.

Bridget Allison closed the panel by saying that she relies far too much on volunteered information for NT security and that formalizing the information would be very welcome. She also stated that more public documentation would be welcome.

This session, like many others, had Microsoft employees speaking openly, and not defensively. Only the issue of Kerberos 5 standards was not openly discussed – not surprising, because Microsoft will be creating its own "standard" in this area.

Something that did not come up in this workshop, but did in the previous one, has an impact on security. Tod Needham, while speaking on a panel about the need for NT source, said that the RPC mechanism in NT has some very serious security flaws and that he doesn't want too many people poring over the source until these problems has been fixed. I found that interesting, because I had heard similar rumors about vulnerabilities from the hacker community this summer. Let's hope Microsoft handles this one, or perhaps we may yet see the "Microsoft worm."

TECHNICAL SESSION

Administration III: Miscellaneous

Summary by Steve Hillman

This session saw three papers presented. The first, "Implementing Large NT Networks in a Heterogeneous Computing Environment," was given by Freddie Chow of the Stanford Linear Accelerator Center (SLAC).

SLAC's network consists of roughly 700 UNIX boxes, 700 Macs, 200 VMS stations, 200 Win3.1 stations, and 500 NT stations. The NT boxes were all installed very recently.

SLAC's computing support group quickly discovered that the existing support model, consisting mostly of casual employees hired by each department, was not adequate for managing an increasingly complex network. Their new support model calls for centralized server and service support and centralized coordination of client support with departmental system administrators. They also do centralized purchases of software for the site, allowing them both to get discounted software prices and to at least control somewhat the software that goes on client machines.

SLAC has established several sitewide computing policies: they've decided to implement a single sitewide NT logon – any user can log on from any workstation. There's also to be no Windows 95 on site. Additionally, several domain policies were established, including:

1. All NT workstations must belong to a domain. A catchall domain was set up for workstations in departments that don't have enough machines to justify their own domain.
2. For departments running their own domains, there must be adequate hardware for the Domain Controllers.
3. Every domain must have a PDC and two BDCs. They must be on a UPS and located in a secure area.
4. Domain system administrators are to be technically proficient and must carry pagers

The computing support group realized it couldn't ignore Win95 completely. A large research site such as SLAC will have visiting researchers with notebook computers. To deal with this, they developed a Win95 policy that states:

- Win95 is not supported.
- Visitors need to report their machines to departmental sysadmins upon their arrival.
- Departmental sysadmins will check the machine for proper network configuration.
- Visitors should revert their machines to original config at departure if possible and reclaim the IP address

When looking at management software, Chow compared Microsoft's SMS to NICE/NT – a management package developed at CERN. NICE/NT offered support for Novell and NDS as well as NT, but SMS offered remote application installation and OS updates. Because SLAC has no Novell, they decided to use SMS. In talking with other sites running SMS and in testing they did themselves, they found that a practical configuration for an SMS machine was a dual Pentium Pro-200 station with 256Mb of RAM.

SLAC is also running NTrigue to allow non-NT users to run NT applications.

Till Poser of DESY asked how SLAC was generating its NT workstations. Chow replied that they were using cloning. He didn't elaborate, but in light of the SID issues that came up at the conference earlier, he may have to rethink that process.

Ken Rudman elaborated on the dual-Pentium requirement of SMS. SMS uses a SQL server to store all its information. Normally, the SMS server and SQL server are run on two separate machines, and the SQL server can benefit from the dual-Pentium configuration. SMS by itself should not need such a powerful machine.

Will Figueroa of Mentor Graphics asked for some more information on the NTrigue server hardware and how many users it can handle. The server at SLAC is a dual Pentium-Pro with 512Mb RAM and 100baseT. They run very few people on it and so haven't really stressed it, but according to colleagues at Stanford University, a similarly configured server was running with 32 concurrent users without significant slowdown.

Steve Hearn asked what SLAC's SMS hierarchy was. Chow responded that currently they're only deploying it on a small scale with just a single site. Bill Evans of Nortel commented that they'd installed a number of standalone SMS sites and then installed a master on top to pull all the sites together into a hierarchy.

Till Poser asked what trust relationships were in place for the 12 domains that they had. Chow said they had a single-master domain model with a single domain for accounts. The 12 domains are resource domains with between 50 and 150 workstations each.

The second paper, "Effective Usage of Individual User NT Profiles with Software Distribution," was presented by Mark Murray and Troy Roberts of Interprovincial Pipe Line Inc. (IPL). IPL runs an oil pipeline that stretches across most of Canada. Along the pipeline, they have over 100 field offices, roughly 30 of which are manned. Their WAN reaches most offices, and they have a mixture of Win3.1/95, Windows NT, VMS, and UNIX workstations spread through their offices.

In devising their infrastructure, they kept several design goals in mind:

- Universal logon (A user should be able to log on from anywhere and in some cases, from any platform type.)
- Service basis (Services should be network based, rather than platform or host based – i.e., all services available to all platforms. Relocation of services should be transparent to the user.)
- Centralized administration
- Automated patch and software distribution (independent of platform type)
- Consistent naming across all platforms
- One workstation per office (Applications for other OSs should be available through Telnet, NTrigue, etc.)
- Standardized software loads (Wherever possible, all machines run exactly the same software. This makes machine restoration easy.)

To accomplish most of these goals, IPL used NT profiles for users. They divided the profile into a user profile, which contains all of the settings relevant to a particular user, and a machine profile, which contains machine and global software settings. They also added the concept of a file space to the profiles. Profiles normally deal only with registry settings, but adding file space allowed them not only to specify what registry settings would be set, but what files should be present. This was done using combinations of shares to have appropriate files (both user data and application binaries) accessible to the user at a consistent location.

Software installation is accomplished by splitting software packages into two separate packages – a user package and a machine package. The user package gets installed into a user's profile and follows that user around. The machine package gets installed into each machine that requires that application and stays with the machine.

Not all machines run every application, and not all users need every application. To manage all of this, a database was set up. Every time a user logs in to a workstation, the database is queried to determine what apps the user should have and what apps the machine is allowed to run. The database has a Web-based front end that allows help desk staff to assign applications to users and machines. New assignments take effect the next time a user logs in or a machine is logged into.

This is an extremely sophisticated software installation infrastructure. For a lot more detail on how it works and what was done, refer to their paper.

Questions started with the most common question of the conference: how, if at all, did they integrate passwords between NT and UNIX boxes? Unfortunately, they haven't tackled this problem yet. Passwords are still stored in two separate places, and no synchronization is done.

Todd Williams asked them to comment on the amount of time it took to implement this project and whether it was justified. Roberts estimated that it had taken them about 18 months from conception to completion. During the actual rollout at the end, they had a staff of about 12 working on the project. He said that this project has drastically reduced their total cost of ownership. It's extremely easy to add new machines and to support existing machines because the machines are always in a known state.

Ian Alderman commented that NT normally leaves behind the profile of the last user who logged in. He wondered if they'd seen this problem. Roberts responded that, in general, this hasn't been a problem. For one thing, users tend to stick to using their own machines (for now) and, for another, because their profiles completely rebuild the machine's environment when a user logs in, any old information is completely flushed out.

An audience member commented on a product called P-synch, which runs on a Windows machine and allows users to change their passwords on every system at once. It's available from M-tech, <http://www.m-tech.ab.ca/psynch>.

Rob Elkins asked exactly how applications are downloaded to machines when users log in to different machines. Murray responded that, in general, they try to avoid this. Most applications are preinstalled on systems. Apps such as Office are installed on every machine. Other large packages, such as AutoCAD, are installed on selected machines and will not be installed onto other machines. Small applications are Just-In-Time installed if a machine doesn't have them and a user who's supposed to have them logs in. This installation is done at login time.

John Holmwood of NOVA Gas Transmission asked how a user's profile gets updated when new applications are installed if the user is currently logged in.

Murray responded that, because applications are split into two separate packages, machine-specific part of the package can be installed immediately, and the user-specific package can be installed the next time the user logs in.

Someone asked what tools were used to implement this system. Roberts said that WinInstall was used to create the separate packages, and Perl was used behind the scenes on the servers.

The last paper of the session, entitled "System Administration of Scientific Computing on Windows NT," was presented by Remy Evard from Argonne National Lab.

Evard and his group got the idea that they might be able to build a supercomputing architecture around a farm of Pentium Pro-based PCs for a fraction of the cost of the existing supercomputers. They based this on tests that show that Pentium Pro-200-based systems compare favorably to supercomputers when matched processor to processor.

Their plan, quite simply, was:

- Build a cluster of NT workstations.
- Figure out how to make it work.
- Turn it over to the computer scientists to port tools.
- Turn it over to the physicists to put it to work.

To start with, they built a cluster of several Pentium Pro 200 PCs interconnected with 100baseT. The main problem they faced was getting remote access to the PCs. NT is designed to be used from the console, and in their environment, that was rarely practical.

They found that Ataman's RSH offered a reasonable remote-access solution. It still wasn't perfect – it didn't load in the user's registry settings to allow for personalized drive mappings and such – but it permitted users to schedule jobs remotely.

They also set up a machine to run NTrigue on. They contemplated putting it on each node in the cluster, but that doesn't scale well because it requires a user to make a connection to every node one at a time. Instead, the NTrigue machine serves as a sort of "front end" to the cluster, allowing users who don't have their own NT workstations to access NT resources and build programs.

The cluster has been up for several months now, and they're calling it a qualified success. According to Evard, it works, but it's slow. Although there are still bugs to work out and tools to develop, the hardware is available cheap off the shelf, making expansion of the system relatively painless.

Lyle Meier asked if they'd considered using Timbuctu for remote access. Evard responded that, again, it doesn't scale to hundreds or thousands of nodes. It would be usable to connect to the front end, because NTrigue is being used now.

Till Poser asked if they'd looked at LSF to handle scheduling of jobs. Evard said that they had purposely avoided dealing with scheduling yet. They still hadn't perfected scheduling on their IBM supercomputer, so they didn't want to complicate the NT experiment. When it comes time to look at scheduling, they'll look at LSF as well as others.

Poser also asked Evard to expand on his "it's slow but it works" statement. Evard explained that it appeared to be a software problem involving an intercommunication library that was being used – the code ran faster on a single machine than on two, and on four, it pretty much stopped. They're convinced that interprocess communication will always be the most difficult obstacle to overcome. The Pentium chip itself will have no problems competing on a number-crunching basis.

PRESENTATIONS AND PANEL

Tips, Tricks, and Gurus

Summary by Gus Hartmann

This panel consisted of Mike Carpenter, Mike Frederick, Bruce Schrock, and Paul Pavlov from Pencom and Eric Pearce from O'Reilly & Associates. Mike Carpenter is the manager of the answer desk at Pencom. He has 11 years of UNIX experience. Mike Frederick urged everyone in attendance to obtain the Resource Kits. Bruce Schrock has been with Pencom for one-and-a-half years, and has been working with Windows NT since version 3.1. Paul Pavlov is from a mostly UNIX background. Eric Pearce is a half-time network administrator and half-time author at O'Reilly and Associates with 12 years of UNIX experience, mostly on Suns, and 2 years of Windows NT experience.

After this brief introduction, the panel began taking questions.

When browsing, the client has a fixed limitation on the number of shares displayed. Why?

Each subnet has a master browser. Every machine broadcasts its presence. The master browser picks up the broadcast and synchronizes with the Primary Domain Controller. Based on the version of the client and the master browser, there are some limitations on the number of shares displayed.

Is there a scriptable setup for Windows NT print servers?

It can and has been done, but the script is proprietary code and can't be released.

Is there an equivalent to sudo for Windows NT?

No.

Are there any books or tutorials analogous to "UNIX for VMS users" to assist UNIX users when dealing with Windows NT?

Not in any single unified source.

Is there a passive way to set system time based over a network?

No, there are only active commands such as `net time /set /y`.

IP anomalies are occurring while multi-homing Windows NT. What can be done to prevent or minimize this?

Don't multi-home Windows NT unless absolutely necessary.

INVITED TALK

MS/Systems Management Server (SMS)

David Hamilton,
Microsoft Corporation

Summary by Steve Hillman

The final session of the workshop was an invited talk by Microsoft on SMS. The session nearly didn't happen – news came down on Thursday that the speaker had had to cancel. This did not go over well with the audience, and one has to wonder what transpired at Microsoft when, on Friday, the audience was informed that the session would happen after all.

The talk was presented by David Hamilton, who is the product manager for Systems Management at Microsoft. David emphasized that SMS is just one product that the Systems Management group puts out, but it is their key package. SMS is currently at release 1.2

SMS is primarily focused on three key areas: Inventory, software distribution, and diagnostics. Inventory deals with the hardware and software on each workstation. Hamilton said that SMS is very heavily inventory focused. In his opinion, you have to have good inventory tracking to be able to perform either of the other two functions. The second function, software distribution, is the function most people buy SMS for. It allows you to distribute applications and OS updates to every workstation on the network. The last function, diagnostics, allows you to diagnose remote workstations. One of the most used tools is the remote-control tool that allows you to see and control a workstation's desktop.

Although SMS has a Windows NT focus, it will also manage Netware, LAN Manager, LAN Server, MS-DOS, Windows 3.1, Windows 95, Macs, and OS/2 clients.

SMS is geared toward enterprise networks with hundreds or thousands of workstations. It is designed to be distributed – multiple-site organizations can have a site server at each location. Each site-server becomes the point of communication both into that site and back to the master server. Site servers do not have to be Windows NT servers. If an organization has Netware servers already deployed, SMS has agents that can run on those servers and provide basic SMS functionality.

SMS is also bandwidth aware. It can be configured to use only a certain percentage of available WAN bandwidth for software distribution, diagnostics, or management. It can also be configured to use that bandwidth only at certain times during the day.

Hamilton admitted that SMS 1.2 is weak in the area of remote software installation (a key feature that many people buy it for), but Microsoft has recently released a couple of add-ins that promise to greatly improve it. The first product, Package Command Manager (PCM), runs as a service on each NT workstation and allows installations that must run with administrative privileges to run regardless of what user (if any) is logged in at the workstation.

The second add-on is the SMS Installer, which is an extremely sophisticated package installer that uses snapshooting to assemble packages. This is useful for applications that either aren't designed for remote installation or can't be automatically installed at all. Essentially, a snapshot is taken of a "test" machine before installation; then the application is installed manually, and another snapshot is taken afterwards. The differences are compared, and a package containing all of them, in both files and registry settings, is assembled. A script is also put together to automatically install the package at each workstation.

It's claimed that SMS Installer makes intelligent decisions about what does and does not need to be installed at each workstation (in the cases where remote workstations are configured differently than the "test" machine). By doing before and after snapshots of the workstation, the installer provides the capability to do an uninstall or even a full rollback in case of difficulty. SMS Installer also supports signing of packages to increase security.

On the subject of NT 5.0, Hamilton explained that, although it makes a lot of advancements in the area of manageability, NT 5.0 does not overlap much with SMS's capabilities. Both inventory management and diagnostics are strictly the domain of SMS. Where there is some overlap is with automated software installation. NT 5.0 includes a product currently named Darwin that attempts to simplify rollout of applications in an enterprise network. The SMS Installer has been designed to interact with Darwin, handing off to Darwin on the workstations when it's present.

Microsoft's Systems Management group is spearheading another new development: Web Based Enterprise Management (WBEM). This is somewhat of a bad title, because WBEM has nothing to do with the Web. Essentially, WBEM is to systems management what SNMP is to network management, although WBEM will not be constrained to just systems. The goal is to provide a common layer and interface for getting at information about a system. That information could be software config info, user info, hardware info, network stats, or performance data. It doesn't matter; the WBEM layer will bring it all together and translate it if necessary. WBEM is being included in both Windows 98 and NT 5.0 and promises to greatly improve the manageability of these platforms by products such as SMS.

Kristina Harris asked Hamilton to clarify what software is available today and what platforms it runs on. SMS is currently at revision 1.2 and will run on either NT 3.51 or NT 4.0. Several add-ons have been produced; most are bundled in the SMS Service Pack 2. The two add-ins, PCM and SMS Installer, are separately downloadable. All of these add-ins work under both NT 3.51 and 4.0.

An audience member asked for clarification on whether SMS needed SQL Server and a PDC or not. Hamilton said that SMS requires SQL Server (either 6.0 or 6.5). It does not work with any other database software. SMS must run on a Primary or Backup Domain Controller; however, it does not have to run on the same machine as the SQL Server (in fact, for performance reasons, it's better if it does not).

Another audience member asked if documented APIs are available for WBEM. There are, and Hamilton referred him to a Web site where it's all available: <http://wbem.freerange.com>.

Someone asked if SMS Installer's rollback capability consumes a lot of disk space. Hamilton said that it does, because it has to save a copy of everything it replaces. But the installer includes sophisticated scripts that allow you to uninstall by walking backwards through the script. This isn't the same as a full rollback, but it uses little or no disk space.

Remy Evard asked what resources are available for understanding and successfully deploying SMS and how users can get feedback to the development team on areas that need improvement. Hamilton said that there's a lot of information online on the Microsoft Web page, including planning guides for rolling out SMS and MS Office using SMS. See <http://www.microsoft.com/smsmgmt> For feedback, there are two mailing lists, <smswish@microsoft.com> and <manageit@microsoft.com>, for SMS-specific and general management suggestions, respectively.

Someone asked if Microsoft intends to support UNIX with SMS. Hamilton said that they worked with DEC quite a while ago to develop UNIX management. They ended up handing the work over to DEC, and it's been released under the title Assetworks.

Jonathon Ott of HP asked if there is a way to do SMS's remote control from the Windows 95 platform. Hamilton said that a company called Computing Edge makes an SMS add-on that allows the SMS admin program to run under Windows 95.

An audience member noted that SMS lacks support for "distribution by courier." [*Editor's note:* Courier may be the Tivoli Systems software distribution product.] Hamilton said they are working on functionality that will allow you to transfer a software package on a CD-ROM via courier and just send the commands over the WAN to install the package. This functionality will appear in the next full release of SMS.

Derek Simmel of CERT asked what security information the inventory manager could gather and also what security measures are being used on software distribution. Hamilton, admitting that he isn't a security expert, said that the inventory manager can gather any information that a process running on the remote workstation has access to – such as registry settings or file information. As far as the software distribution is concerned, he said they're not introducing any new holes, but are just using the existing network structure that's already in place (although he had said earlier that SMS Installer supports packet signing).

David Blank-Edelman from Northeastern University commented that he'd been doing systems management for at least 15 years and had come across a huge wealth of information, some of which was in the form of conference proceedings and such. He wondered if Microsoft had access to that information and was using it in its design philosophies. Hamilton said that he spends a lot of his time talking with customers to see what they want in the products, but he had no idea whether Microsoft had any of those proceedings.

Chris Kulpa of Michigan State University asked how the HKEY_LOCAL_USER hive in the NT registry gets updated when a package is installed and a user is not logged on. Hamilton said that currently it isn't. What really needs to happen is that packages need to be split apart into a user section and a system section and each section installed separately. The user section would be installed the next time the user logs on. Microsoft plans to have this in the next release. Note that this is the same functionality that IPL wrote (see the Administration III session summary on page 62).

Bill Evans asked what can be done about software that requires software keys or serial numbers to be entered at the workstation as it's installed. Hamilton said it's possible to put the numbers required into the registry as part of the distribution and then pull out the desired number as part of the install.

Rik Farrow said that most UNIX administrators are nervous about turning an installation script with administrative privileges loose on a system. Hamilton replied that you can "test drive" a script first and get a printout of exactly what changes the script will make to the system without it actually making the changes. This allows you to avoid things like installation of back doors.

T.J. Fisk of Qualcomm commented that his company has a software librarian who handles distribution of software to individuals. Currently the packages are mailed out, taking a week or more to be delivered. He wanted to know if it is possible in SMS to automate this, making it easy to deliver a single package to a single person. Hamilton responded that currently this is possible with SMS, but it's difficult. There are rule-based algorithms for deciding what machines to deliver software to, but one needs a lot of training to write those rules. With the introduction of Microsoft Management Console in NT 5.0, this should get a lot easier.

Scott Fadden of Sequent Computers asked if there is a database schema available for the SMS database. Hamilton said this is probably the single most frequent request they get, but they can't provide one because their database changes too much from release to release. Instead, they supply a utility called SMS Views that provides a series of ODBC interfaces into the database.

An audience member pointed out that throughout the conference, there's been a strong interest in getting more command line support into NT. He asked if SMS would support more command line functionality. Hamilton responded that they're working on that slowly, but they still won't have everything accessible from the command line by NT 5.0.

This marked the last session of the workshop. Phil Scarr concluded with a few words. When he asked who would attend a workshop like this next year, virtually all hands went up. The program committee had already agreed that another workshop would be a good idea.

The Large Installation System Administration of Windows NT Conference is scheduled for August 5-7, 1998, in Seattle, WA.

Common Themes and Observations (from Phil Scarr's Closing Remarks)

NT is far too/more complicated (than UNIX)

Management demands rapid deployment

No good management tools

NT is too "closed"

Support departments are given far too few resources to manage NT

There are not enough NT "experts" yet

Proposed NT enterprise infrastructure doesn't scale

NT is becoming instantly mission critical

User namespace integration is hard

Microsoft is suprisingly willing to listen

Integration issues are becoming better understood

Samba is cool

Everybody is having the same problems I am

Announcement and Call for Participation

2nd USENIX Windows NT Symposium

August 3-5, 1998
Renaissance Madison Hotel
Seattle, Washington

Sponsored by USENIX, the Advanced Computing Systems Association

Important Dates

Paper submission deadline: *March 3, 1998*
Break-out session proposal deadline: *March 20, 1998*
Notification of acceptance: *April 3, 1998*
Final papers due: *June 18, 1998*

Program Committee

Symposium Co-Chairs
Susan Owicki, *InterTrust Technologies Corporation*
Thorsten von Eicken, *Cornell University*

Steering Committee
Ed Lazowska, *University of Washington*
Michael B. Jones, *Microsoft Research, Microsoft Corporation*
Margo Seltzer, *Harvard University*

Committee
John Bennett, *Rice University*
Pei Cao, *University of Wisconsin, Madison*
Brad Chen, *Harvard University*
Anton Chernoff, *Digital Equipment Corporation*
Jim Gray, *Microsoft Bay Area Research Center*
Carl Hauser, *Xerox Palo Alto Research Center*
Michael B. Jones, *Microsoft Research, Microsoft Corporation*
Ed Lazowska, *University of Washington*
Jane Liu, *University of Illinois at Urbana-Champaign*
Nick Vasilatos, *VenturCom Corporation*
Werner Vogels, *Cornell University*
Stephen R. Walli, *Softway Systems, Inc.*
Rumi Zahir, *Intel Corp.*

The 2nd USENIX Windows NT Symposium is a forum for researchers actively using or planning to use Windows NT to discuss ideas and share information, experiences, and results. The first USENIX Windows NT Workshop attracted over 300 participants actively engaged in product development, industry research, and academic research using the Windows NT platform. The symposium will follow the model established by the 1997 workshop; the renaming merely acknowledges the size of the event.

The symposium will include technical presentations of submitted papers, invited talks, break-out sessions for highly interactive focused discussions, informal Birds-of-a-Feather sessions, and tutorials. While proceedings will be published, the primary purpose of the symposium is to facilitate 2.5 to 3 days of useful interaction among the participants — not to provide yet another specialized systems publication venue.

We encourage anyone using or adopting Windows NT as their research base to participate.

Topics

We are soliciting presentations and break-out session proposals covering a broad range of topics, including, but not limited to:

- Applications
- Manageability
- Security
- Availability
- Performance and scalability
- Networking and distributed systems
- File and database systems
- Graphics
- User interfaces
- Hardware architectures
- Programming environments
- Tools and utilities
- Porting and integration into existing environments

Proposals for Technical Presentations

We are soliciting presentations of concepts, research, and experiences relevant to researchers using Windows NT. Participants interested in making technical presentations should email a three to six page short paper to *usenix-nt-submissions@usenix.org* by March 3, 1998. Accepted papers will receive up to 10 pages in the proceedings. The email containing your submission must have a subject line reading "NT symposium submission" and must begin with the following information in this format:

Title: (title of submission)
Authors: (names of all authors)
Contact: (primary contact for submission)
Address: (contact's full postal address)
Phone: (contact's telephone number)
Fax: (contact's fax number if available)
E-mail: (contact's email address — very important)

Submissions must be in Microsoft Word (Word '97 preferred over Word '95), HTML, or FrameMaker format and should be encoded for transport with either the uuencode or the MIME base64 encoding. Filenames used in your submission should be based on your last name, e.g. *smith.doc* and *smith.html.* Receipt of submissions will be acknowledged by return email within one week; if an acknowledgment is not received, please send email to *usenix-nt-questions@usenix.org.*

Note that the USENIX organization, as well as most conferences and journals, requires that papers be unique, i.e., not be submitted to more than one conference or publication. All submissions are held in the strictest confidence prior to publication in the conference proceedings, both as a matter of policy and as protected by the U.S. Copyright Act of 1976.

Proposals for Break-Out Sessions

The symposium will include scheduled break-out sessions on topics of interest to different sub-groups. Examples of kinds of break-out sessions include working groups, panels, and sets of demonstrations; others are also possible. If you want to see a particular topic or problem addressed at the symposium, break-out sessions are intended to provide you with a vehicle to make it happen. Final selection will be based in part on interest indicated on registration forms. Proposals for break-out sessions should be emailed to *usenix-nt-submissions@usenix.org* with the subject line "NT symposium break-out session proposal" by March 20, 1998. Please describe the format of the proposed session and indicate if you are willing to help organize it.

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are very informal gatherings of attendees interested in a particular topic to be held in the evenings. BoFs may be scheduled in advance by emailing *usenix-nt-submissions@usenix.org* with the subject line "NT symposium BoF proposal". They may also be scheduled at the symposium.

Works-in-Progress Session

The symposium will include a session of short presentations on works-in-progress. Information on submitting works-in-progress session proposals will be made available on the symposium Web site.

Usage Abstracts

All symposium participants without an accepted position paper will be required to submit a one page abstract or summary at the time that they register describing what they are doing or considering doing with Windows NT.

These abstracts will be made available on the symposium web site before the symposium and will also be distributed in paper form to attendees. The abstracts are intended to facilitate communication among attendees, helping people find others with similar interests or problems. Send usage abstracts to *usenix-nt-abstracts@usenix.org.*

Registration

Materials containing all details of the technical and symposia programs, registration fees and forms, and hotel information will be available by May, 1998. If you wish to receive the registration materials, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest CA 92630
714.588.8649
Fax: 714.588.9706
Email: *conference@usenix.org*
URL: *www.usenix.org*

Questions: If you have questions about the symposium that are not address by the symposium web site, send mail to *usenix-nt-questions@usenix.org.*

About USENIX

USENIX is the Advanced Computing Systems Association. Since 1975, USENIX has brought together the community of engineers, system administrators, and technicians workin on the cutting edge of the computing world. For more information about the Association and its activities:

URL: *www.usenix.org*
Email: *office@usenix.org*
Fax: 510.548.5738
Phone: 510.528.8649

Announcement and Call for Participation

Large Installation System Administration of Windows NT Conference (LISA NT)

August 5–7, 1998
Renaissance Madison Hotel
Seattle, Washington

Sponsored by USENIX, the Advanced Computing Systems Association
Co-sponsored by SAGE, the System Administrators Guild.

Co-located with the 2nd USENIX Windows NT Symposium
August 3–5, 1998

Important Dates

Referreed paper submissions: *March 3, 1998*
All other submissions: *March 10, 1998*
Notification to authors: *March 31, 1998*
Camera-ready papers due: *June 18, 1998*
Registration material available: *Mid-May 1998*

Conference Organizers

Co-chairs

Remy Evard, *Argonne National Laboratory*
Ian Reddy, *Simon Fraser University*

Program Committee

Michael Carpenter, *Pencom Systems Administration*
Kendall Martin, *Microsoft*
Chris Rouland, *Lehman Brothers*
Phil Scarr, *Netscape Communications*
Mark Verber, *WebTV*

Overview

Success brings challenge. Windows NT's success brings with it the challenge of administering and integrating large scale installations of NT hosts. The Large Installation System Administration of Windows NT conference, LISA NT, is a forum to bring system administration professionals together to discuss workable solutions to the issues of scaling the NT environment.

LISA NT will enable dialogue with peers and experts in our field. We invite you to submit technical track papers and proposals to enhance the invited talks, panels, Birds-of-a-Feather sessions, and Work-in-Progress reports. We also invite proposals for tutorials and demos of products and solutions.

Please review this call for participation, make a submission, and join us in futhering LISA NT's position as the premiere conference for system administrators of large scale NT environments.

We look forward to your participation.

Topics

LISA NT is about creating a community of system administrators to discuss common problems and share solutions learned from the management of thousands of NT machines and the integration of NT into complex computing environments. To facilitate these discussions, we encourage you to consider these topics:

Management

- Large-scale NT management solutions and experiences
- Locally developed administration tools and techniques
- Successful approaches to centralized, remote, and distributed management
- Large-scale software and operating system deployment
- Reduction of total cost of ownership

Sharing and integration

- Integration of NT and UNIX environments
- Shared NT solutions
- Remote access
- Laptops and PDAs

Tools, experiences, and issues

- Performance tuning and measurement
- Central name service, browsing, and e-mail management
- Software license, deployment, and upgrade management
- Security issues, including policies, education, response, and fixes
- The registry, user accounts
- Backups and restore

What to Submit

The most important aspect of the LISA NT Conference is the sharing of information. With this in mind, the program committee seeks submissions from the Windows NT system administration community in the following formats.

Referred Papers

We seek papers relating work of general interest to system administrators of Windows NT, particularly technical papers that reflect hands-on experience or describe implementable solutions.

Submissions will be judged on the quality of the written submission and whether or not the work advances the art and science of NT system administration.

A paper submission should:

- contain a short abstract
- include an outline of the paper
- conform to the "How and Where..." instructions below

If you have a completed paper, you may submit it instead of the abstract and outline.

For detailed author instructions and a sample abstract, you will find instructions at our Web site
www.usenix.org/events/lisa-nt98/guidelines.html
or send email to
lisa-nt-98-guidelines@usenix.org

Authors of an accepted paper must provide a final paper for publication in the conference proceedings. At least one author of each accepted paper will present the paper during the technical track of the conference. We also ask that, if possible, copies of presentation slides be made available for publication.

Conference proceedings containing all refereed papers will be distributed to attendees and will also be available from USENIX once the conference ends.

Note that the USENIX organization, as well as most conferences and journals, requires that papers be "unique", i.e., not be submitted to more than one conference or publication. All submissions are held in the strictest confidence prior to publication in the conference proceedings, both as a matter of policy and as protected by the U.S. Copyright Act of 1976.

Invited Talks/Panel Discussions
If you have a presentation that is not suitable for a technical paper submission, please submit a proposal for an invited talk or panel. The prosposal should include:

- an extended outline of the talk or panel topic and format
- a description of your qualifications to present the topic
- list the likely participants on the panel
- conform to the "How and Where..." instructions below

Acceptance will be based upon the general applicability of the topic and on availability of time in the program.

Works-in-Progress Reports (WIPs)
WIPs are short talks that introduce new or ongoing work. If you have work you would like to share or a cool idea that is not quite ready to be published, a WIP is for you.

To submit a WIP:

- include a description of the problem and your (possibly incomplete) solution
- if necessary, include an explanation of why the problem you are addressing is interesting
- conform to the "How and Where..." instructions below

Acceptance will be based upon the applicability and scalability of the proposed solution.

How and Where to Send Submissions

Please email your submission to
lisa-nt-98-submissions@usenix.org
in any one of the following formats:

- Plain text with no extra markup
- Postscript formated for 8.5" × 11" page
- Microsoft Word
- RTF
- HTML

A cover letter with the following required information in the format below must be included with all submissions:

- Authors: Names and affiliation of all authors
- Contact: Primary contact for the submission
- Address: Contact's full postal address
- Phone: Contact's telephone number
- Fax: Contact's fax number
- Email: Contact's e-mail address
- URL: For all speaker/authors (if available)
- Category: Category of the submission (paper, invited talk, panel, WIP)
- Title: Title of the submission
- Needs: Audio-visual requirements for presentation

If you enclose files as an attachment to your submission, please use MIME encoding.

We will acknowledge receipt of a submission by email within one week.

Tutorials

There will be full and half-day tutorials in all areas and levels of expertise for Windows NT system administrators.

If you are interested in presenting a tutorial at LISA NT please contact the USENIX tutorial coordinator:

Daniel V. Klein
Email: *dvk@usenix.org*
Phone: 412.421.0285
Fax: 412.421.2332

Birds-of-a-Feather (BOF) and Demonstration Sessions

BOF sessions are very informal gatherings of attendees interested in a particular topic. Demonstrations of NT tools, techniques, and products, whether freely available or commercial, are also welcome, though you will have to supply your own equipment. BOFs and demonstrations will be held on Wednesday and Thursday evenings and may be scheduled at the conference or in advance by sending email or phoning the USENIX Conference Office

Email: *conference@usenix.org*
Phone: 714.588.8649

Registration Information

Complete program and registration information will be available mid-May 1998 at the USENIX Web site,
www.usenix.org/events/lisant98/

If you would like to be added to our mailing list, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest,CA USA 92630
Email: conference@usenix.org
Phone: 714.588.8649
Fax: 714.588.9706

About USENIX

USENIX is the Advanced Computing Systems Associaton. Since 1975, USENIX has brought together the community of engineers, system administrators, and technicians working on the cutting edge of the computing world.

For more information about USENIX, SAGE, and our events, visit our Web site,
www.usenix.org.

USENIX®, the Advanced Computing Systems Association

how I spent my summer vacation

Or, Escape from the Evil Empire

by Brian Dewey

Brian Dewey is a second-year graduate student in computer science at the University of Washington.

<dewey@cs.washington.edu>

Fatigued by three quarters of classwork as a first-year graduate student at the University of Washington, my ears perked up when I heard through the department grapevine that Rob Short, group manager of NT infrastructure software, was looking for a summer intern. I wanted to spend my summer outside of the department, so this seemed like a fantastic opportunity. When I met with Rob a few weeks later, things only looked better – he told me that, among other things, his group could use help with Wolfpack, NT's clustering technology. I couldn't have hoped for a more fascinating project. So I submitted myself to the Microsoft interview process – that alone could be the subject of an essay – and began my internship in June.

I spent my first day as a Microsoft intern riding around in a bumper car while attempting to throw a wiffle ball through a hole in a basketball backboard. When I wasn't doing that, I was busy drinking free Red Hook beer or playing the Elvira pinball game with an inexhaustible supply of somebody else's quarters. I even won a door prize – a maroon insulated backpack with a full set of picnic supplies and "Microsoft" emblazoned on one corner. Not a bad way to begin. This wasn't the beginning of a summer internship; it was the start of summer camp!

Microsoft employees certainly like to play. The day-to-day games are on a much smaller scale than the wiffle-ball party; vintage arcade games lurk in the halls of building 26, inviting developers to put down their keyboards. (Battle Zone, an Atari arcade game from 1980, became my preferred break during long compiles.) I saw people take the time to do some of the oddest – and frequently, geekiest – things. For instance, a fellow intern brought in thirty pounds of dry ice during his last week and spent an hour distracting conscientious programmers (such as myself!) by filling offices with the cold smoke.

Unfortunately, the games weren't the only thing that filled the hours during the first part of my internship. Rob had told me that the fax group needed someone for a quick file format conversion project. As I got more involved, I found out that it involved a lot more than file format conversion and that it would be anything but quick. I consoled myself with the fact that it was a boring but necessary job, but it turned out to be a disheartening experience.

I spent a depressing amount of time doing metaprogramming. Not surprisingly, I faced the learning curve of a new environment; however, on top of that, my work lived in a limbo between three distinct programming groups. I spent too much time bouncing from one group to another and attempting to piece together a clear picture of what I was supposed to do. It didn't help that the project was about as interesting as oatmeal. About halfway through my internship, I had a serious crisis of motivation: if nobody in the three groups wanted to understand how their components interacted, why should I – an intern and therefore without the amazingly motivating stock options – go through the effort of synthesis?

During this period, the hours in Redmond seemed to drag on without end, and I had to reevaluate my first summer camp impressions. Although Microsoft seems a high-tech playground at times, the employees earn their playtime through an extreme devotion to their work. The expectation of long hours was vaguely frightening. It was so common for people to work into the evening – especially as a deadline approached – that Microsoft provided free dinners to the NT development group through the Marriott-run cafeteria in building 26. The programmer I worked with the most mentioned that he had a habit of coming in to work at least one day on the weekend. Without the distractions of meetings, it was the best time to program. And on the Friday before Labor Day, there was an espresso cart outside the mailroom on the second floor; a sign stated that there would be free espresso drinks those Saturday and Sunday afternoons. (New slogan: "Microsoft: we put the 'labor' back in 'Labor day.'")

But like conscientious programmers are supposed to do, I prevailed over the boredom and finished my project in the fax group. I'm in the final part of my internship as I write this, working on a project with the Wolfpack group, and I've experienced a shocking reversal of morale. I'm getting the precise combination of detective work, problem solving, and video game playing (during compiles, of course) to remind me why I love the programmer's life. I'm starting to work longer hours without even noticing, and part of me regrets the fact that the end of my internship is rapidly approaching.

I've consequently revised my opinion of Microsoft again. The company doesn't expect long hours from its employees. However, it does expect the employees to love their work, whatever facet of the production cycle it may fall into. The long hours come as a consequence, not as a goal. I think this is an accurate characterization of the industry. After all, stereotypical programmers are obviously devoted to their work – late nights in the lab, meals by the keyboard – and this stereotype carries more truth than many of us might care to admit. Of course, the increasing need to rush a product to market stands in opposition to the ideals of people who work for the love of it and want to take the time to do the job right. But again, the entire industry faces this problem. After spending a summer at Microsoft, I realize it no longer looks like summer camp or an Evil Empire. It's just a software company.

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

AUTOMATIC INFORMATION SERVER

If you do not have access to the Web, finger <info@usenix.org> and you will be directed to the catalog which outlines all conferences, activities, and services.

PGP INFORMATION

All correspondence to:
Key ID: D6C05861 1996/11/11
USENIX 1997 <pgp@usenix.org>
6118D4DBE52612EB 8031C57DAE9AC168

Master key, for signatures only:
Key ID: 2FEA2EF1 1996/04/08

USENIX master key <not-for-mail>:
DBA 7509966E48AA9 80B2D9E2FEDA005E

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are December 9, 1997 and February 3, 1998.

USENIX

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES